Freie Universität Berlin

Fachbereich Mathematik und Informatik

Mixed-Integer Optimization for Loopless Flux Distributions in Metabolic Networks

Hannah Marie Troppens
Matrikelnummer: 5039637
hannah.troppens@fu-berlin.de

22. March 2024

Abstract

Zusammenfassung

Contents

Abbreviations

1 Introduction

2 Mathematical Optimization

link to other sections

Optimization appears in various disciplines. In timetabling, we seek a schedule at school such that teachers are assigned to students at a given time [17]. Of course different requirements have to be satisfied. Each teacher and student should have only one class at a time. The classes should not be too big and the teachers are only allowed to teach a limited number of hours per week. Another example for an optimization problem is clustering, where we want to divide a set of data points into subsets that share similarities [24]. Optimization also arises in nature. Animals adapt and optimize their behaviour through learning [4]. And even the behaviour in cells optimizes biological objectives such as maximizing growth or minimizing energy usage [36]. Optimization in cells is the topic of this thesis and we will see in Section 3 how to use mathematical optimization to answer questions arising in systems biology.

book Palsson, objective functions in practice; plus page 5 for evolution

In mathematical optimization, the goal is to find an optimal solution that respects requirements that arise in our application. In order to represent a problem in practice by a mathematical model, we need to formulate an objective. An objective in timetabling could be minimizing the number of used classrooms, and in clustering an objective is to minimize the difference between the points in the same cluster. Additionally, we need to identify entities of the system. In timetabling, we would have students, teachers, rooms and time as different entities in the model. We also need to identify constraints in our system. Once the objective, the entities and the requirements of a system are identified, it can be expressed mathematically. The resulting model is an *optimization problem*, and we can use optimization algorithms to compute solutions. We define an optimization problem as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}) \tag{1a}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \tag{1b}$$

where f is the *objective function*, g_i are the *constraint functions* and $\mathbf{x}$ are the *decision variables*. The *feasible region* is the set of points that respect the constraints. An optimization problem with a constrained feasible region is often also called *constrained optimization problem* in contrast to an *unconstrained optimization problem*. A *solution* is a vector $\mathbf{x}$ that lies in the feasible region. An *optimal solution* $\mathbf{x}^*$ is a

solution with the smallest objective value. The *objective value* of $\mathbf{x}^*$ is the value of the objective function evaluated at $\mathbf{x}^*$. If a problem has no solution, it is said to be *infeasible*. A problem is *unbounded* if solutions exist, but the objective value can be arbitrarily small. One can maximize a function f by setting the objective function to $\min -f(\mathbf{x})$.

Depending on the type of the objective function and the type of constraints, optimization problems are divided into different classes. The classes relevant for this thesis are linear programs, mixed-integer programs and disjunctive programs[1]. Different algorithms can be used to solve optimization problems, which depend on the problem structure.

2.1 Preliminaries and Notation

use variables consistently; check when to use if and when iff; which words to write in italic

1. A vector $\mathbf{v} = (v_1, ..., v_n) \in \mathbb{R}^n$ is identified as a column vector and printed in bold. $\mathbf{v}^\intercal$ transposes $\mathbf{v}$ into a row vector. The inner product of two vectors $v, w \in \mathbb{R}^n$ is $\mathbf{v}^\intercal\mathbf{w} = \sum_{i=1}^n v_i w_i$. $\mathbf{v} \leq \mathbf{w}$ denotes element-wise inequality. $\ln(\mathbf{v})$ denotes the element-wise natural logarithm. The 1-vector is written as $\mathbf{1} := (1, 1, ..., 1) \in \mathbb{R}^n$ the 0-vector as $\mathbf{0} := (0, 0, ..., 0) \in \mathbb{R}^n$. The *support* of $\mathbf{v}$ is the set of indices i with $v_i \neq 0$ and is denoted by $\text{supp}(\mathbf{v})$. $\text{sign}(\mathbf{v})$ is the element-wise sign function applied. The concatenation of two vectors $\mathbf{x}$, $\mathbf{y}$ is denoted as $(\mathbf{x}, \mathbf{y})$.

2. The entry in row i and column j of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is denoted as $a_{i,j}$. The i-th row is $\mathbf{a}_{i,*}$ and the j-th column $\mathbf{a}_{*,j}$.

   update in entire section, often one index used for row of matrix

   The zero matrix is denoted by $\mathbf{0}_{m,n}$ with m rows and n columns. $\text{diag}(\mathbf{v})$ is the quadratic matrix with $\mathbf{v}$ on the diagonal and 0 for all other entries.

3. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an invertible matrix: $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The matrix $\mathbf{A}^{-1}$ is the inverse matrix of $\mathbf{A}$. $\mathbf{I}_n$ is the $n \times n$ identity matrix.

4. A *linear combination* is defined as $\sum_{i=1}^m \lambda_i \mathbf{x}_i = \lambda_i x_1 + ... + \lambda_m x_m$, where $\lambda_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$. A line going through a point $\mathbf{x}$ generated by $\mathbf{r} \in \mathbb{R}^n$ is the set

[1] Program here does not mean a computer program. The term was coined in the 1950s and referred to planning in a military context. See [42] for more detail.

$\{\mathbf{x} + \lambda\mathbf{r} | \lambda \in \mathbb{R}\}$. A *line segment* is a subset of a line defined on the interval between $l \in \mathbb{R}$ and $u \in \mathbb{R}$: $\{\mathbf{x} + \lambda\mathbf{r} | \lambda \in \mathbb{R},\ \lambda \in [l, u]\}$.

5. A *basis* B of a vector space V is a set of vectors $(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n)$ that are linearly independent and every $\mathbf{v} \in V$ can be written as a linear combination of vectors in B.

6. The *nullspace* of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as $\text{null}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$.

7. A set $C \subseteq \mathbb{R}^n$ is *convex* if for any two points $\mathbf{x}, \mathbf{y} \in C$ the line segment between $\mathbf{x}$ and $\mathbf{y}$ is in C. The *convex hull* of a set X is the smallest convex set that contains all points in X and denoted as $\text{conv}(X)$. It is a set of convex combinations such that all points x_i in X can be represented, where a *convex combination* is a linear combination with $\lambda_i \geq 0$ and $\sum_{i=1}^{m} \lambda_i = 1$.

8. The set $\{\mathbf{x} \in \mathbb{R}^n | \boldsymbol{\alpha}^\intercal \mathbf{x} = \boldsymbol{\beta}\}$ is a *hyperplane*, where $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{\beta} \in \mathbb{R}$. The set $\{\mathbf{x} \in \mathbb{R}^n | \boldsymbol{\alpha}^\intercal x \leq \boldsymbol{\beta}\}$ is a *half-space*. A *polyhedron* $P = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is the intersection of a finite number of half-spaces, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Hyperplanes, half-spaces and polyhedra are convex. A point $\mathbf{x} \in P$ is an *extreme point* or *vertex* if it cannot be represented as a convex combination of any set of other points in P. A polyhedron is also called *polytope* if P can be written as the convex hull of the extreme points of P.

   verify

   add reference

9. A set $C \subseteq \mathbb{R}^n$ is a *cone* if $\lambda\mathbf{x} \in C$ for any $\mathbf{x} \in C$ and $\lambda \geq 0$. A *conic combination* is a linear combination with $\lambda_i \geq 0$. C is *pointed* if it contains no line and the extreme point is called *apex*. A nonzero vector $\mathbf{r} \in \mathbb{R}^n$ is a *ray* of C iff $\{\mathbf{x} + \lambda\mathbf{r} | \lambda \geq 0\} \in C$ for any $\mathbf{x} \in C$. Ray $\mathbf{r}$ is an *extreme ray* if it cannot be represented by a conic combination of other rays in C. A cone is *polyhedral* if the number of extreme rays is finite. A cone is *simplicial* if it has n extreme rays.

   verify

   The *conic relaxation* of an extreme point $\mathbf{x}$ of a polyhedron P is a cone with apex $\mathbf{x}$, and the extreme rays are the half-spaces of P intersecting at $\mathbf{x}$.

10. Let $S \in \mathbb{R}^n$ be a nonempty set and $\mathbf{x}$ a point in S. $\mathbf{x}$ is in the *interior* of S, denoted as $\text{int}(S)$, if there exists an $\epsilon > 0$ such that any point in the ball

centered at $\mathbf{x}$ with radius ϵ is contained in S. $\mathbf{x}$ is on the *boundary* of S, denoted as $\text{bd}(S)$, if it is not in $\text{int}(S)$. S is *closed* if the boundary of S is contained in S, S is said to be *open* otherwise. This also holds if the boundary is the empty set and therefore $\mathbb{Z}$ is closed.

should be definition for open, and closed is complement of open set

2.2 Linear Programming

A *linear program* (LP) is an optimization problem with a linear objective function and linear constraints.

LP:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{2a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{2b}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{2c}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. It is assumed that the number of constraints m is greater than the number of variables n. The linear inequalities define a polyhedron. If an optimal solution exists, there exists an optimal solution at one of the vertices.

double check with reference

In that case, the LP can have exactly one optimal solution, or multiple optimal solutions if an entire edge or face

hyperface?

is optimal. An LP has no optimal solution if it is infeasible or unbounded. Problem (2) is said to be in *inequality form*.

As an example, we want to solve the following optimization problem:

$$\max_{x,y} \quad y \tag{3a}$$

$$\text{s.t.} \quad 0 \leq x \leq 3 \tag{3b}$$

$$0.5 \leq y \leq 4 \tag{3c}$$

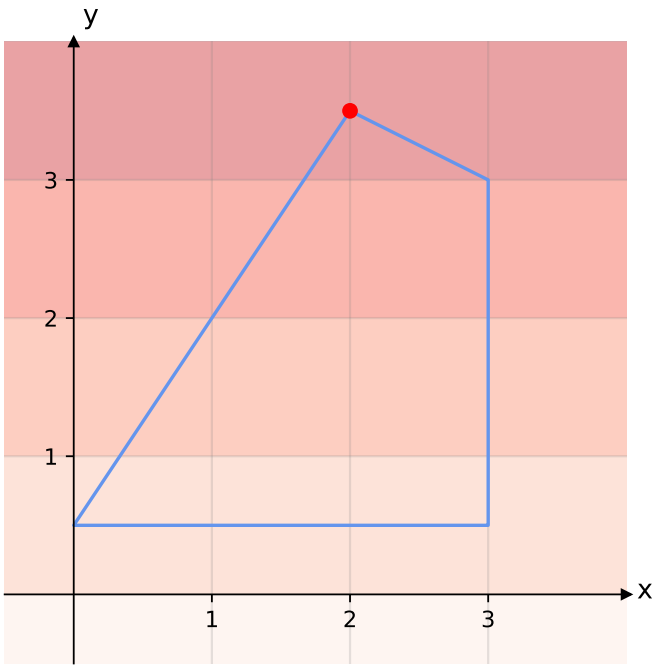
$$y \leq 1.5x + 0.5 \tag{3d}$$

$$y \leq -0.5x + 4.5 \tag{3e}$$

$$x, y \in \mathbb{R} \tag{3f}$$

The decision variables x, y are continuous. We want to find a solution (x^*, y^*) with maximal y-value such that the constraints are respected. The problem is a linear program and can be written as Problem (2) by simple linear transformations. If we want to write Problem (3) as a minimization problem, the objective becomes $-y$. If we visualise the example (Figure 1), we see that there exists one optimal solution located at $(2, 3.5)$. Usually, we are interested in problems in higher dimensions and it is not possible to solve them visually.

Figure 1: visualization of Problem (3)



The set of feasible solutions contains all the points in the interior or on the boundary of the polytope (blue line segments). The optimal solution is the point in the feasible region with the biggest y-value (red point). The isolines indicate the function value of the points.

remove grid?

An LP is said to be in *standard form* if it is of the form:

$$\min_{x} \quad \mathbf{c}^\intercal \mathbf{x} \tag{4a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{4b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{4c}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Problem (2) can be written in standard form by adding one *slack variable* per inequality to write it as equality [30]. $\mathbf{a}_{i,*}^\intercal \mathbf{x} \leq b_i$ becomes $\mathbf{a}_{i,*}{}^\intercal \mathbf{x} + s_i = b_i$ with $s_i \geq 0$. Each variable x_i that can be negative is replaced by $x_i^+ - x_i^-$, where $x_i^+ \geq 0$ and $x_i^- \geq 0$ [30].

A vertex of the feasible region is also called *basic feasible solution*. Suppose we have an LP in polyhedral form as in Problem (2). Let $\mathbf{x} \in \mathbb{R}^n$ be a basic feasible solution. There exist n constraints that are a basis defining $\mathbf{x}$.

mention proof or cite

B is the set of constraint indices that are in the basis. Any variable $x_i \in B$ is a *basic variable*. A variable $x_i \notin B$ is a *nonbasic variable* and $x_i = 0$ [42].

2.2.1 Solving LPs

There exist several algorithms to solve linear programs. The *ellipsoid method* is the first algorithm that was proven to have a polynomial running time in the worst case. However, in practice, other algorithms outperform it [42]. Another class of algorithms are *interior-point methods*. Some interior-point methods have a polynomial running time in the worst case and are often used in practice. Interior-point methods start with a feasible solution in the interior of the feasible region and approach the optimum without stepping outside the feasible region [42].

Another algorithm that is relevant in practice is the *simplex algorithm*. It is based on the fundamental property of LPs that the optimum is located at one of the vertices of the feasible region. In *phase I* of the algorithm, a vertex of the feasible region is computed. In *phase II*, the algorithm moves from vertex to vertex along edges of the feasible region until an optimum is found. The *pivot rule* determines to which vertex the algorithm moves next. There exist several pivot rules, however for all of them, there are families of problem instances on which the number of pivot steps needed grows exponentially. The worst case running time on some instances is in conflict with the observed polynomial running time in practice. Studying the simplex method with *smoothed analysis*, which tries to close this gap, shows a polynomial running time of the simplex method. In smoothed analysis, a small noise is added to the entries of a fixed instance and afterwards worst-case analysis is performed on the perturbed instance [21, 14].

2.2.2 Optimality and Duality

To prove that a solution is optimal, we can use the *Karush-Kuhn-Tucker*-conditions (KKT-conditions) and see the relation between the primal and the dual problems in LPs. An optimization problem can be written as an unconstrained problem by augmenting the objective function with a weighted sum of the constraints [8]. The resulting function is known as the *Lagrangian* function. The Lagrangian of an LP in

standard form is [30]:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{c}^\intercal \mathbf{x} - \sum_{i=1}^{n} \lambda_i x_i - \sum_{i=1}^{m} \nu_i (A_{i,*}^\intercal \mathbf{x} - b_i) \tag{5}$$

where $\lambda_i \geq 0$ and $\nu_i \in \mathbb{R}$ are called *Lagrange multipliers* or *dual variables*. As the objective function of linear programs is convex, the KKT-conditions are a proof for global optimality of a solution. We obtain the optimality conditions captured in Theorem 1.

Theorem 1 (Optimality conditions for LPs) *A solution* $\mathbf{x}^*$ *is optimal if there exist vectors* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\nu}$ *that satisfy the following conditions [30]:*

1. $\boldsymbol{\lambda} + \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{c}$ *(stationarity)*
2. $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$ *(primal feasibility)*
3. $\mathbf{x} \geq \mathbf{0}$ *(primal feasibility)*
4. $\boldsymbol{\lambda} \geq \mathbf{0}$ *(dual feasibility)*
5. $\mathbf{x}^\intercal \boldsymbol{\lambda} = \mathbf{0}$ *(complementary slackness)*

We call a linear program in standard form as in Problem (4) the *primal problem* ($\mathcal{P}$). The associated *dual function* of the primal problem is [2]:

MB: shouldn't the conditions $x \geq 0, \lambda \geq 0$ appear somewhere?

$$\begin{aligned}
q(\boldsymbol{\lambda}, \boldsymbol{\nu}) &= \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\
&= \min_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x} - \boldsymbol{\lambda}^\intercal \mathbf{x} - \boldsymbol{\nu}^\intercal (\mathbf{A}\mathbf{x} - \mathbf{b}) \\
&= \min_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x} - \boldsymbol{\lambda}^\intercal \mathbf{x} - \boldsymbol{\nu}^\intercal \mathbf{A}\mathbf{x} + \boldsymbol{\nu}^\intercal \mathbf{b} \\
&= \min_{\mathbf{x}} (\mathbf{c}^\intercal - \boldsymbol{\lambda}^\intercal - \boldsymbol{\nu}^\intercal \mathbf{A})\mathbf{x} + \boldsymbol{\nu}^\intercal \mathbf{b} \\
&= \min_{\mathbf{x}} \mathbf{x}^\intercal (\mathbf{c} - \boldsymbol{\lambda} - \mathbf{A}^\intercal \boldsymbol{\nu}) + \mathbf{b}^\intercal \boldsymbol{\nu} \\
&= \begin{cases} \mathbf{b}^\intercal \boldsymbol{\nu} & \text{if } \mathbf{c} - \boldsymbol{\lambda} - \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{0} \\ -\infty & \text{otherwise} \end{cases}
\end{aligned}$$

check if it has to be inf

Let $\mathbf{x}^P$ be a feasible solution to the primal problem, and let $(\boldsymbol{\lambda}^D, \boldsymbol{\nu}^D)$ be a feasible solution to the dual problem. If the dual function is bounded we know that $\mathbf{c} - \boldsymbol{\lambda}^D =$

$\mathbf{A}^\intercal \boldsymbol{\nu}^D$ and we know that $\mathbf{x}^P$ satisfies $\mathbf{A}\mathbf{x}^P = \mathbf{b}$. The function value $q(\boldsymbol{\lambda}^D, \boldsymbol{\nu}^D)$ is a lower bound on the objective value of the primal problem [2]:

$$\mathbf{b}^\intercal \boldsymbol{\nu}^D = (\mathbf{A}\mathbf{x}^P)^\intercal \boldsymbol{\nu}^D = \mathbf{x}^{\mathbf{P}\intercal} \mathbf{A}^\intercal \boldsymbol{\nu}^D = \mathbf{x}^{\mathbf{P}\intercal}(\mathbf{c} - \boldsymbol{\lambda}^D) = \mathbf{c}^\intercal \mathbf{x}^P - \boldsymbol{\lambda}^{\boldsymbol{D}\intercal} \mathbf{x}^P \leq \mathbf{c}^\intercal \mathbf{x}^P$$

The tightest bound maximizes $\mathbf{b}^\intercal \boldsymbol{\nu}$. Formulated as a linear program we obtain the *dual problem* ($\mathcal{D}$):

$$\max_{\boldsymbol{\nu}, \boldsymbol{\lambda}} \quad \mathbf{b}^\intercal \boldsymbol{\nu} \tag{6a}$$

$$\text{s.t.} \quad \boldsymbol{\lambda} + \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{c} \tag{6b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{6c}$$

or rewritten without the slack variables $\boldsymbol{\lambda}$ [30]:

$$\max_{\boldsymbol{\nu}} \quad \mathbf{b}^\intercal \boldsymbol{\nu} \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{A}^\intercal \boldsymbol{\nu} \leq \mathbf{c} \tag{7b}$$

Let p^* be the objective value of an optimal solution for the primal problem and d^* the objective value of an optimal solution to the dual problem. We have seen that d^* is a lower bound on p^* which is known as *weak duality.*

Theorem 2 (Strong Duality) *Given a primal $\mathcal{P}$ and corresponding dual program $\mathcal{D}$, exactly one of the following is true [30]:*

1. *$\mathcal{P}$ and $\mathcal{D}$ both have at least one optimal solution. If p^* is the objective value of an optimal solution to $\mathcal{P}$ and d^* is the objective value of an optimal solution to $\mathcal{D}$, then $p^* = d^*$.*
2. *Either $\mathcal{P}$ or $\mathcal{D}$ is unbounded and the other is infeasible.*
3. *$\mathcal{P}$ and $\mathcal{D}$ both are infeasible.*

For linear programs, *strong duality* holds. A proof for Theorem 2 can be found in [2]. Property (1) of Theorem 2 can be used to prove the optimality of a solution.

We have seen how to derive the dual problem of an LP in standard form. We do not require an LP in standard form to derive the corresponding dual problem. For

example, the dual problem of Problem (2) is:

$$\max_{\boldsymbol{\mu}} \quad \mathbf{b}^\intercal \boldsymbol{\mu} \tag{8a}$$

$$\text{s.t.} \quad \mathbf{A}^\intercal \boldsymbol{\mu} = \mathbf{c} \tag{8b}$$

$$\mu_i \geq 0 \tag{8c}$$

2.3 Mixed-Integer Programming

Many problems in practice cannot be formulated by only using linear constraints and continuous decision variables. It is often required that a subset of variables is discrete. A *mixed-integer program* (MIP) is an optimization problem with a linear objective function, linear constraints and a subset of integer variables:

MIP:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{9a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{9b}$$

$$\mathbf{x} \in \mathbb{Z}^{|J|} \times \mathbb{R}^{n-|J|} \tag{9c}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. The set J contains the indices of integer variables.

Let us reuse the LP example in Section 2.2 and add integrality constraints on the decision variables x and y:

$$\max_{x,y} \quad y \tag{10a}$$

$$\text{s.t.} \quad 0 \leq x \leq 3 \tag{10b}$$

$$0.5 \leq y \leq 4 \tag{10c}$$

$$y \leq 1.5x + 0.5 \tag{10d}$$

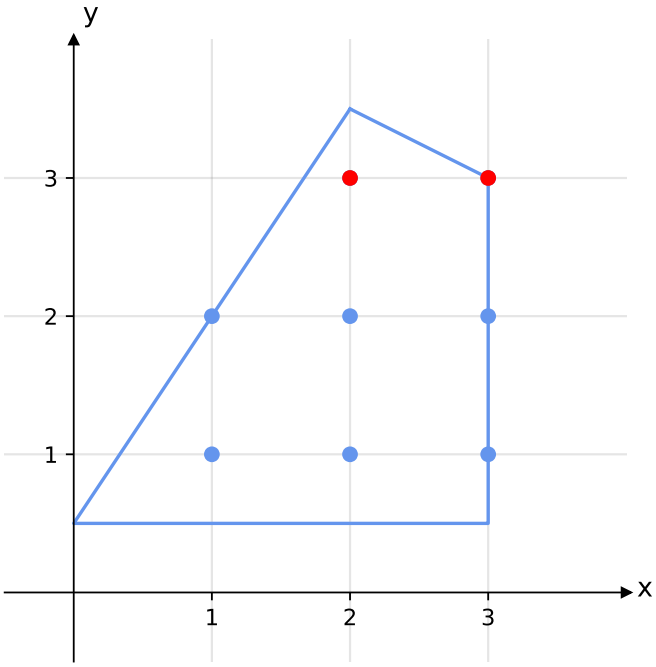
$$y \leq -0.5x + 4.5 \tag{10e}$$

$$x, y \in \mathbb{Z} \tag{10f}$$

Looking at the visualisation of the problem (Figure 2), we see that the optimal solution of the LP $x^{LP} = (2, 3.5)$ is no valid solution for the MIP formulation. The points $(2, 3)$ and $(3, 3)$ are the optimal solutions.

The MIP formulation enables us to model much more complex problems. Apart from incorporating discrete quantities, it is possible to capture Boolean expressions.

Figure 2: visualisation of Problem (10)



The feasible points of the MIP are the integer points respecting the polyhedral constraints (blue points). The set of feasible solutions to the relaxed LP is are all the points in the interior or on the boundary of the polytope (blue line segments). An optimal solution is a point in the feasible region with the biggest y-value (red point). The optimal solutions are at $(2, 3)$ and $(3, 3)$.

Suppose we want to model the *indicator constraint* $z = 1 \implies \mathbf{a}^\intercal \mathbf{x} \leq b$. We can reformulate the constraint with a linear constraint using the *big-M method*:

$$\mathbf{a}^\intercal \mathbf{x} \leq b + M(1 - z)$$

If $z = 1$ the constraint is enforced. In the case $z = 0$, the value of M has to be larger than $\mathbf{a}^\intercal \mathbf{x} - b$ for any $\mathbf{x}$ such that the constraint is inactive [2].

Solving MIPs

Solving MIPs is much more complicated than solving LPs, because it is not guaranteed that if an optimal solution exists, it is at one of the vertices. In general, solving MIPs is $\mathcal{NP}$-hard. A problem is $\mathcal{NP}$-hard if it requires at least as much time to solve the problem as any other problem that is $\mathcal{NP}$-complete. $\mathcal{NP}$-complete problems are $\mathcal{NP}$-hard and a solution is verifiable in polynomial time. In complexity theory, problems are defined as decision problems.[12] The decision problem of Problem (9) would be "Is the feasible region nonempty?", which is $\mathcal{NP}$-complete. As finding an optimal solution is not easier than finding any solution, it follows that MIPs are $\mathcal{NP}$-hard. [10]

fix citations

Instead of solving the MIP directly, one can solve a sequence of *LP relaxations*: the integrality constraints are ignored and Constraint (9c) becomes $\mathbf{x} \in \mathbb{R}^n$. Let z^{LP} be the objective value of an optimal solution $\mathbf{x}^{LP}$ of the LP relaxation and z^* the objective value of an optimal solution $\mathbf{x}^*$ to the MIP problem. We know that:

$$z^{LP} \leq z^*$$

One common approach for solving MIPs is the *branch-and-bound* algorithm [10]. The idea is to generate a branch-and-bound tree starting with the solution to the LP relaxation $\mathbf{x}^{LP} \in \mathbb{R}^n$ at the root node. A variable x_i that is fractional in $\mathbf{x}^{LP}$ and violates the integrality constraint is selected as *branching variable*. We divide the search space P by creating two child nodes. In one x_i has to be larger or equal to the rounded up value and the feasible region becomes $P \cap \{x_i \geq \lceil x_i^{LP} \rceil\}$. In the other child node, x_i can take at most the rounded down value of x_i^{LP} and the feasible region of the subproblem is $P \cap \{x_i \leq \lfloor x_i^{LP} \rfloor\}$. The solution with the smallest objective value respecting the MIP formulation is called *incumbent* and the objective value is denoted by z^{INC}. We continue branching until all variables in J are integral, a subproblem is infeasible or if a node can be *pruned*. The optimal solution in each node i is bounded by the objective value of the relaxed solution $z_{(i)}^{LP}$. If $z_{(i)}^{LP}$ is greater or equal to z^{INC}, node i is cut off the tree.

Another approach to solve MIPs is the *cutting plane* algorithm [10]. The LP relaxation can be very weak. As we are dealing with a linear objective, we could get the optimal MIP solution easily if we had access to the *integer hull*: $\text{conv}(P \cap \mathbb{Z}^n)$. If the LP relaxation does not correspond to the integer hull, one can tighten the LP relaxation by adding *cuts*. A cut is an inequality that does not cut off any feasible solution (see Section 2.5). Given an optimal solution to the LP relaxation $\mathbf{x}^{LP}$ that violates at least one integer constraint, one separates $\mathbf{x}^{LP}$ from the hull with a cut. This process is repeated until $\mathbf{x}^{LP}$ is an optimal solution to the MIP.

A combination of the branch-and-bound algorithm and the cutting plane algorithm is the *branch-and-cut* algorithm [10]. The LP relaxation at a node is potentially tightened by adding cuts.

2.4 Disjunctive Programming

Often, the requirements of a problem are complex, and linear constraints are not sufficient for the mathematical program. With Boolean expressions, more complicated

relationships between variables can be captured without using the big-M formulation. A *disjunctive program* (DP) is an optimization problem with linear constraints, continuous variables and logical constraints:

DP:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{11a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{11b}$$

$$\bigvee_{i \in Q_j} \{\mathbf{d}^{(i)\intercal}\mathbf{x} \leq d_0^i\} \quad \forall j \in S \tag{11c}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. $\mathbf{d}^{(i)} \in \mathbb{R}^n$, $d_0^i \in \mathbb{R}$ and S is the set of disjunction indices. As the terms i in each disjunction Q_j are linear, each disjunctive set is a polyhedron. The feasible region is in general non-convex due to the disjunctive constraints [3]. Alternatively, the disjunctions can be captured by Boolean variables $Y_{ij} \in \{true, false\}$, where Y_{ij} corresponds to the i-th disjunct in disjunction j:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{12a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{12b}$$

$$\bigvee_{i \in Q_j} \begin{bmatrix} Y_{ij} \\ \{\mathbf{d}^{(i)\intercal}\mathbf{x} \leq d_0^i\} \end{bmatrix} \quad \forall j \in S \tag{12c}$$

$$\Omega(Y) = true \tag{12d}$$

As an example, we want to solve the following optimization problem:

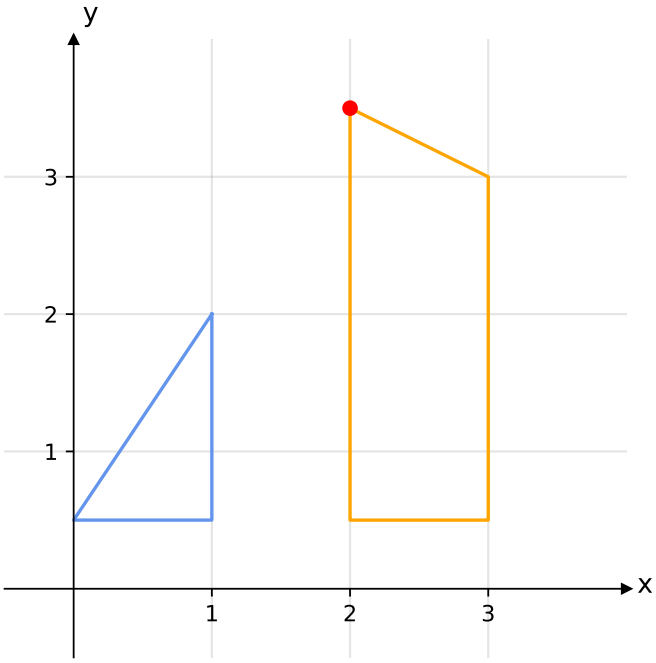
$$\max_{x,y} \quad y \tag{13a}$$

$$\text{s.t.} \quad \begin{array}{l} ((0 \leq x \leq 1) \wedge (0.5 \leq y \leq 1.5x + 0.5)) \vee \\ ((2 \leq x \leq 3) \wedge (0.5 \leq y \leq -0.5x + 4.5)) \end{array} \tag{13b}$$

$$x, y \in \mathbb{R} \tag{13c}$$

The feasible region is the union of two polyhedra and no longer convex. The optimal solution is the point with maximal value in either of the polytopes and located at $(2, 3.5)$ which we see in the visualization (Figure 3).

Figure 3: visualization of Problem (13)



The set of feasible solutions is no longer convex and is the union of the polytope P_1 (blue line segments) and the polytope P_2 (orange line segments). The optimal solution is the point in the feasible region with the biggest y-value (red point).

Solving DPs

A disjunctive model can be formulated as mixed-integer program and solved by corresponding techniques (see Section 2.3). Disjunctions can be expressed by linear constraints and integer variables by using the big-M method. Suppose we have a disjunction with k terms:

$$(\mathbf{a}_1^\intercal \mathbf{x} \leq b_1) \vee (\mathbf{a}_2^\intercal \mathbf{x} \leq b_2) ... \vee (\mathbf{a}_k^\intercal \mathbf{x} \leq b_k)$$

We use the binary variable y_i and enforce with the constraint $y_1 + y_2 + ... + y_k \geq 1$ that at least one of the k terms is true. If $y_i = 1$, term $(\mathbf{a}_i^\intercal \mathbf{x} \leq b_i)$ is enforced. See Section 2.3 on how to express indicator constraints with linear constraints and binary variables. Often, the M constant has to be large to be inactive if $y_i = 0$. However, a large M leads to a weaker LP relaxation which impacts the running time of the Branch-and-Bound algorithm [2].

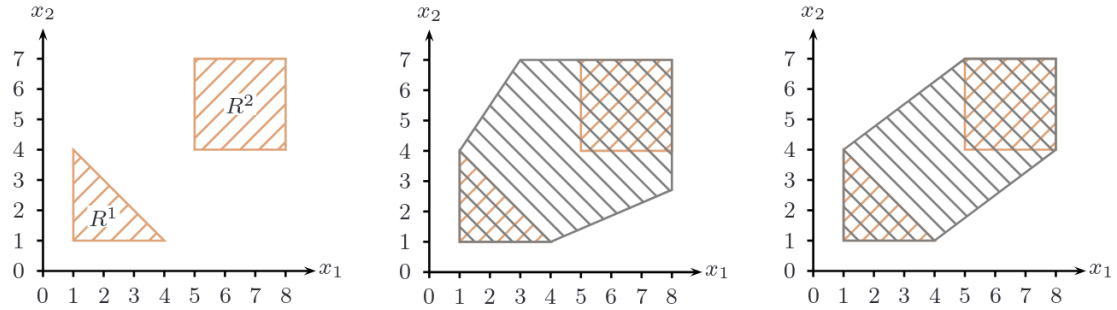
Another possibility is to convexify the feasible region and solve the resulting linear program. The idea is to build the convex hull of the union of polyhedral points in a higher dimension. The optimal solution at a vertex of the convex hull is optimal for the disjunctive program, as we are dealing with linear constraints. For an example

on how to write a disjunctive program using the big-M or the hull reformulation, we refer to [34]. Figure 4 shows the 2D-projection of the big-M reformulation and the convex hull reformulation of a disjunctive program.

how to solve if corner can be infeasible?

update example and create own figure for hull and big M diagram

Figure 4: taken from [1]



The feasible region of a disjunctive program (left), the approximation of the region using the big-M reformulation (center) and the convex-hull reformulation (right).

2.5 Cutting Planes

Let us consider a mixed-integer problem (Problem (9)) with decision variables $\mathbf{x}$. A *cutting plane* or *cut*, parameterised by $\boldsymbol{\alpha} \in \mathbb{R}^{n+1}$, is an inequality that when added to the MIP does not remove any feasible solution, and is defined as:

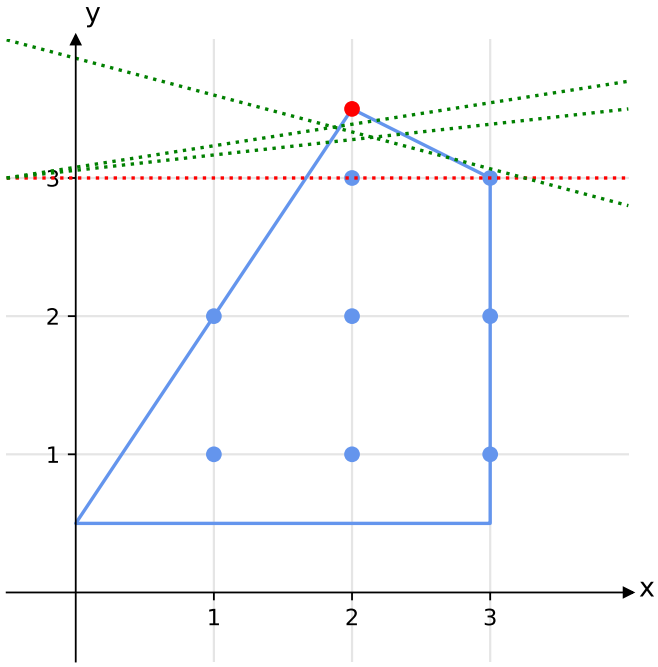
$$\sum_{i=1}^{n} \alpha_i x_i \leq \alpha_0 \quad \forall \mathbf{x} \in P \cap S \tag{14}$$

where $\alpha_i \in \mathbb{R}$, P is the polyhedron defined by the inequality in Constraint (9b), and S is the set points satisfying Constraint (9c). Usually, cutting planes are used to cut off solutions to the relaxed problem that are infeasible in the original problem.

Figure 5 shows a visualisation of Problem (10) in which the optimal solution to the relaxed problem $\mathbf{x}^{LP}$ is not feasible in the MIP. $\mathbf{x}^{LP}$ is separated from the integer hull by cuts.

For a solution to the relaxed problem $\mathbf{x}^{LP}$ with $\mathbf{x}^{LP} \notin P \cap S$, there exist multiple hyperplanes separating $\mathbf{x}^{LP}$ from the actual feasible region. As we want to tighten the search space, we are interested in cuts that cut off many infeasible points at once. There are different scores to estimate the quality of a cut [41] . One scoring

Figure 5: Tightening the relaxed problem of Problem (10) with cuts



The feasible points of Problem (10) are the integer points within the polytope. In the relaxed problem, the integrality constraints are ignored. The optimal solution to the relaxed problem is $\mathbf{x}^{LP} = (2, 3.5)$. As $\mathbf{x}^{LP}$ is not an integer solution, it is cut off by adding cuts. The tightest cut is indicated by the red dashed line.

measure is *efficacy* which is the signed distance from $\mathbf{x}^{LP}$ to the cutting plane:

$$\text{eff}(\boldsymbol{\alpha}, \mathbf{x}^{LP}) := \frac{\boldsymbol{\alpha}^\intercal \mathbf{x}^{LP} - \alpha_0}{\|\boldsymbol{\alpha}\|} \tag{15}$$

However, there is a tradeoff between the quality of a cut and the complexity of generating it.

numerical issues

The cuts relevant for this thesis are explained in detail below.

2.5.1 No-Good Cuts

Given an integer problem P as in Problem (9) with only binary variables: $J = n$ and $x_i \in \{0, 1\}$. Let $\mathbf{x}^{IP}$ a solution to the relaxed integer program, where Constraint (9b) is relaxed. If $\mathbf{x}^{IP}$ is not a feasible solution to P, we want to exclude the solution from the search space. The following *no-good cut* is added to the relaxed IP and

forbids the assignment of integer variables in $\mathbf{x}^{IP}$:

$$\sum_{j \in J : x_j^{IP} = 0} (1 - x_j) + \sum_{j \in J : x_j^{IP} = 1} x_j \quad \leq \quad |J| - 1$$

Such a cut usually tightens the feasible region of P marginally and it would require a large number of no-good cuts to arrive at the integer hull.

2.5.2 Combinatorial Benders' Cuts

Instead of forbidding one assignment of variables as with a no-good cut, with combinatorial Benders' cuts we generate stronger cuts, by identifying the subset of variables that lead to the infeasiblity. This section is based on [9].

Given a problem of the form:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z}} \quad & \mathbf{c}^\intercal \mathbf{x} && \text{(16a)}\\
\text{s.t.} \quad & \mathbf{F}\mathbf{x} \leq \mathbf{g} && \text{(16b)}\\
& \mathbf{D}\mathbf{z} \leq \mathbf{e} && \text{(16c)}\\
& x_j = 1 \implies \mathbf{a}_{i,*}^\intercal \mathbf{z} \leq b_i \quad \forall i \in I && \text{(16d)}\\
& x \in \{0,1\}^n && \text{(16e)}\\
& z_i \in \mathbb{R} && \text{(16f)}
\end{aligned}$$

As the objective does not depend on $\mathbf{z}$ and as the decision variables $\mathbf{x}, \mathbf{z}$ are independent apart from the indicator constraints, we can split the problem into a *master problem* (MP) and a *subproblem* (SP). Constraint Problem (16) (d) can be reformulated using the big-M method: $\mathbf{A}\mathbf{z} \leq \mathbf{b} + M(\mathbf{1} - \mathbf{x})$.

MP:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathbf{c}^\intercal \mathbf{x} && \text{(17a)}\\
\text{s.t.} \quad & \mathbf{F}\mathbf{x} \leq \mathbf{g} && \text{(17b)}\\
& \mathbf{x} \in \mathbb{R}^n && \text{(17c)}\\
& x_j \in \{0,1\} && \text{(17d)}
\end{aligned}$$

The master problem ignores the constraints on $\mathbf{z}$. The subproblem depends on a solution $\mathbf{x}^{MP}$ to the master problem.

SP:

$$\min_{\mathbf{z}} \quad 0 \tag{18a}$$
$$\text{s.t.} \quad \mathbf{Dz} \leq \mathbf{e} \tag{18b}$$
$$\mathbf{a}_{i,*}^{\intercal}\mathbf{z} \leq b_i + M(1 - x_j^{MP}) \quad \forall i \in I \tag{18c}$$

If solution $\mathbf{x}^{MP}$ leads to a feasible subproblem, $\mathbf{x}^{MP}$ is an optimal solution to Problem (16). If the problem is infeasible, $\mathbf{x}^{MP}$ is not a feasible solution for the original problem. In that case we want to add a cut that removes $\mathbf{x}^{MP}$ from the search space. Suppose we have access to a *minimal infeasible subsystem* (MIS) C that is an inclusion-wise minimal set of row indices corresponding to the constraints in the subproblem that lead to infeasibility. The subproblem can then be written as:

define properly (M.)

$$\min_{\mathbf{z}} \quad 0 \tag{19a}$$
$$\text{s.t.} \quad \mathbf{Dz} \leq \mathbf{e} \tag{19b}$$
$$\mathbf{a}_{i,*}^{\intercal}\mathbf{z} \leq b_i + M_i(1 - x_j^{MP}) \quad \forall i \in C \tag{19c}$$

The *combinatorial Benders' cut* is then:

$$\sum_{j \in C : x_j^{MP} = 0} (1 - x_j) + \sum_{j \in C : x_j^{MP} = 1} x_j \quad \leq \quad |C| - 1$$

A minimal infeasible subsystem can be found by studying the dual problem of the infeasible subproblem. As the subproblem is a linear problem, strong duality holds and an infeasible primal problem implies an unbounded dual problem (see Section 2.2). To derive the dual problem of Problem (18) we write constraints (b) and (c) as one constraint and stack the variables in the vector $\mathbf{y}$: $\tilde{\mathbf{A}}\mathbf{y} \leq \tilde{\mathbf{b}}$. The dual is then:

$$\max_{\boldsymbol{\lambda}} \quad \tilde{\mathbf{b}}^{\intercal}\boldsymbol{\lambda} \tag{20a}$$
$$\text{s.t.} \quad \tilde{\mathbf{A}}^{\intercal}\boldsymbol{\lambda} = \mathbf{0} \tag{20b}$$
$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{20c}$$

As $\boldsymbol{\lambda}$ is a feasible solution the dual problem, the dual problem is unbounded, and there exist multiple feasible solutions. We are interested in a feasible solution with $\boldsymbol{\lambda} \neq \mathbf{0}$ and therefore add the constraint $\tilde{\mathbf{b}}^\intercal \boldsymbol{\lambda} = \mathbf{1}$. Now that the objective function is hidden in the constraints, we can set a different objective function. The linear program to find minimal infeasible subsystems is then:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \sum_i w_i \lambda_i && \text{(21a)} \\ \text{s.t.} \quad & \tilde{\mathbf{A}}^\intercal \boldsymbol{\lambda} = \mathbf{0} && \text{(21b)} \\ & \boldsymbol{\lambda} \geq \mathbf{0} && \text{(21c)} \\ & \tilde{\mathbf{b}}^\intercal \boldsymbol{\lambda} = \mathbf{1} && \text{(21d)} \end{aligned}$$

where w_i is the weight corresponding to the dual variable λ_i. The support of each solution at a vertex of the feasible region of Problem (21) defines a minimal infeasible subsystem. By changing the weights in the objective, several MISs can be obtained for one infeasible MP solution.

2.5.3 Intersection Cuts

Suppose we are given a problem of the form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in S \cap P \end{aligned} \tag{22}$$

where $\mathbf{c} \in \mathbb{R}^n$, P is a polyhedron, and $S \subset \mathbb{R}^n$ is a closed, potentially non-convex set. In the LP relaxation of Problem (22), the constraint set becomes $\mathbf{x} \in P$. However, the relaxation might not be a good approximation of the true feasible region. The polyhedral approximation can be tightened by adding intersection cuts [7].

Let $\tilde{\mathbf{x}}$ be an optimal solution to the LP relaxation that is infeasible in the original problem: $\tilde{\mathbf{x}} \notin S$. The *conic relaxation* at vertex $\tilde{\mathbf{x}}$ is a cone with apex $\tilde{\mathbf{x}}$ and the neighbouring edges are the extreme rays. For a given set S, the convex set C is *S-free* if: $S \cap \text{int}(C) = \emptyset$ [7].

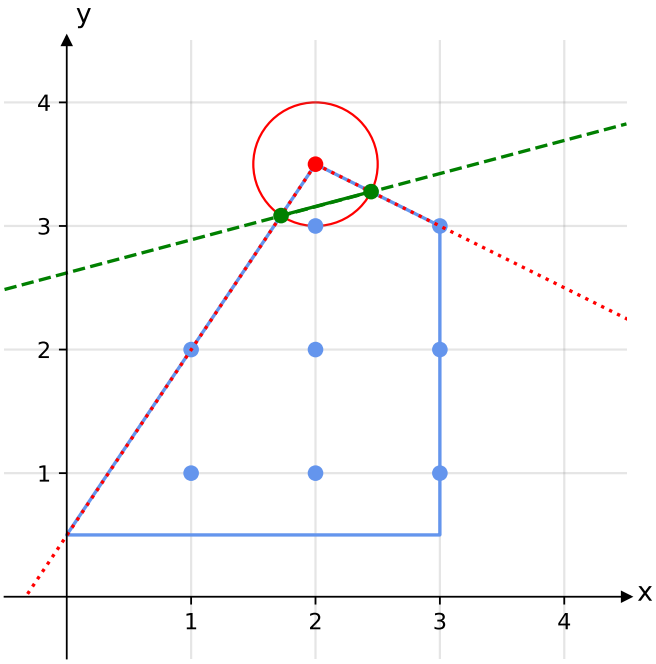
ensure that method section is correct

The hyperplane intersecting the boundary of C and the conic relaxation is the *intersection cut* and when added to the problem will cut off $\tilde{\mathbf{x}}$. Due to convexity of the feasible region and given that the cutoff area lies in C, all feasible solutions remain in the resulting polyhedron [27]. An S-free set C is *maximal* if $C \not\subset C'$ for any S-free set C' [7]. In order to generate deep cuts, the S-free set should be maximal.

As an example, we derive an intersection cut for Problem (10). A visualisation is shown in Figure 6. The optimal solution of the relaxed linear program to Problem (10) is $\tilde{\mathbf{x}} = (2, 3.5)$. The set S contains all integer points: $S = \mathbb{Z}$. A possible S-free set C is a disk where the center is $\tilde{\mathbf{x}}$ and the radius the distance between $\tilde{\mathbf{x}}$ and the closest integer, in our case 0.5. The conic relaxation is the cone with apex $\tilde{\mathbf{x}}$ and the rays correspond to the active constraints at $\tilde{\mathbf{x}}$: $y \leq 1.5x + 0.5$ and $y \leq -0.5x + 4.5$. The intersection between the circle C and the conic relaxation are the points $P_1 \approx (1.7, 3.1)$ and $P_2 \approx (2.5, 3.3)$. The area between $\tilde{\boldsymbol{x}}$ and the line going through P_1 and P_2 is cutoff by adding a constraint to the relaxed LP of Problem (10).

differentiate point from vector in notation?

Figure 6: diagram of intersection cut for Problem (10)



The feasible points of the MIP are the integer points respecting the polyhedral constraints (blue points). The set of feasible solutions to the relaxed LP is are all the points in the interior or on the boundary of the polytope (blue line segments). The optimal solution $\tilde{\mathbf{x}}$ to the relaxed LP is the point in the feasible region with the biggest y-value (red point). A possible S-free set C is a disk with center $\tilde{\mathbf{x}}$ and the radius being the distance between $\tilde{\mathbf{x}}$ and the closest integer (red circle). The conic relaxation are the extreme rays generated by $\tilde{\mathbf{x}}$ which correspond to the two active constraints at $\tilde{\mathbf{x}}$ (is indicated by the dotted, red line). The intersection of the conic relaxation and C are the two points in green. The intersection cut is constructed with the line going through the intersecting points (dashed line in green). The area above the green line will be cut off and $\tilde{\mathbf{x}}$ is no longer a feasible solution.

The conic relaxation at $\tilde{\mathbf{x}}$ is a pointed cone with extreme point $\tilde{\mathbf{x}}$ and can be written as:

$$P' = \{\tilde{\mathbf{x}} + \sum_{j=1}^{n} \lambda_j \mathbf{r}^j : \boldsymbol{\lambda} \geq 0\} \tag{23}$$

or as:

$$P' = \{\mathbf{x} | \tilde{\mathbf{A}}\mathbf{x} \leq \tilde{\mathbf{b}}\} \tag{24}$$

where $\mathbf{r}^j$ are the extreme rays, $\tilde{\mathbf{A}}$ is an invertible $n \times n$ submatrix of $\mathbf{A}$ such that the rows are linearly independent and are a basis for $\tilde{\mathbf{x}}$ [7]. A basis for $\tilde{\mathbf{x}}$ are the nonbasic constraints at $\tilde{\mathbf{x}}$. If a constraint is nonbasic, the corresponding slack variable of the standard form is nonbasic and therefore 0. This implies that the constraint is active.

Going back to Problem (10), the polyhedral presentation of the conic relaxation at $(2, 3.5)$ is:

$$P' = \left\{ (x, y) \middle| \begin{bmatrix} -1.5 & 1 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix} \right\}$$

The inverse of $\tilde{\mathbf{A}}$ is

$$\tilde{\mathbf{A}}^{-1} = \begin{bmatrix} -0.5 & 0.5 \\ 0.25 & 0.75 \end{bmatrix}$$

and we get the extreme rays $r_1 = (0.5 - 2.5)$ and $r_2 = (-0.5, -0.75)$.

After deriving the conic relaxation, we have $\tilde{\mathbf{x}} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$ and $\mathbf{r}^j = -\tilde{\mathbf{A}}^{-1}_{*,j}$ [7]. For each extreme ray $\mathbf{r}^j$ there either exists an intersection with the boundary of C in which case $\lambda^*_j > 0$ is the step length or the extreme ray is contained in C and $\lambda^*_j = \infty$. The *intersection cut* is defined as [7]:

$$\sum_{i=1}^{n} (\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \tilde{b}_i)/\lambda^*_i \leq -1 \tag{25}$$

where $\tilde{\mathbf{a}}_{i,*}$ denotes the i-th row of basic constraints $\tilde{\mathbf{A}}$.

If we reformulate Equation (25), we see that it matches the definition of a cut in Equation (14) [7]:

$$\sum_{i=1}^{n} (\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \tilde{b}_i)/\lambda^*_i \leq -1 \tag{26}$$

$$\sum_{i=1}^{n} ((1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} - (1/\lambda^*_i)\tilde{b}_i) \leq -1 \tag{27}$$

$$\sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \leq -1 \tag{28}$$

$$\sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} \leq -1 + \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \tag{29}$$

$$\alpha_0 = -1 + \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \qquad \alpha_j = \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,j}\mathbf{x}$$

explain why x is cut off

update: basis defining x and basis for P

3 Metabolic Networks

index of v and G differ as they do not have the same length

Computational systems biology studies biological systems and biological networks by building and analysing mathematical models that approximate their behaviour [36]. In this thesis we focus on models at the cellular level. Instead of looking at the function of all molecules in a cell individually, as in molecular biology, systems biology studies the entire system and especially the links between the different cellular components [32]. Looking at the system as a whole leads to a different understanding of the cell, as the behaviour of the cell depends on the interaction of the different components [36] and cannot be explained by only studying the function of components individually.

In many cells, the chemical reactions that take place in an organism, known as *metabolism*, determine the central function of the cell [36]. Understanding the metabolism is therefore necessary in order to understand the cellular behaviour. *Metabolites* are small molecules that are involved in metabolic reactions [36]. We differentiate between *internal reactions*, involving only internal metabolites, and *exchange reactions*. Exchange reactions are pseudo reactions representing the transport of metabolites, where either nutrition is taken up or a product is discarded by the cell [37]

is this true in general or just for COBRA models?

. In a reaction, a set of metabolites called *reactants* are converted into a different set of metabolites called *products*. The *stoichiometry* is the relative number of metabolites in a reaction. *Enzymes* are important for the cell metabolism as they *catalyze* reactions, which means that they accelerate reactions without being consumed by it. Graphs can be used to model and understand cellular behaviour [36]. In a *metabolic network*, the nodes are metabolites and the edges represent reactions. A *genome-scale metabolic model* (GEM) or *enzyme constrained metabolic model* is a graph that contains all known metabolic information of a biological system such as reactions, metabolites and enzymes [33].

The information required to build a GEM comes from a *genome-scale reconstruction*, which is the process of identifying the genome of the organism including different components such as the stoichiometry, reaction direction and the corresponding catalyzing enzymes [37]

about sentence

. Depending on the data and assumptions integrated in a model, a different accuracy

is achieved, and different questions can be answered [37].

In this thesis, we focus on constraint-based approaches, also known as *constraint-based resconstruction and analysis* (COBRA). A constraint excludes biologically unrealistic behavior, such as violating the laws of thermodynamics or the conservation of mass [36]. With constraint-based methods, one can predict the rate of each reaction in the metabolic network [36]. The mathematical representation of a metabolic network is described in Section 3.1. The COBRA models relevant for this thesis are presented in Section 3.2.

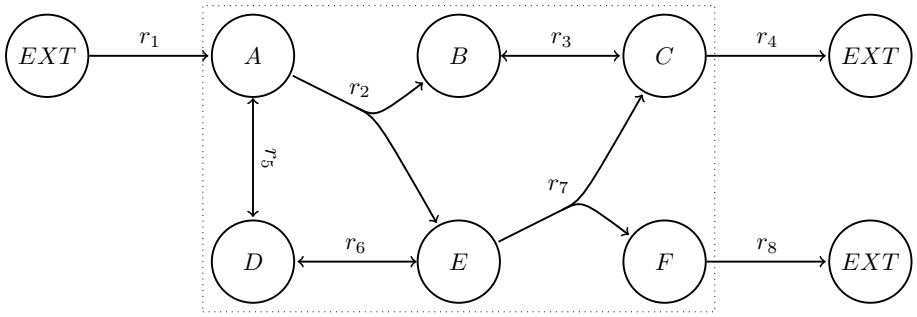
3.1 Mathematical Representation

A *hypergraph* $\mathscr{H}$ is defined by its set of vertices $\mathscr{V}$ and set of hyperedges $\mathscr{E}$. A hyperedge is a pair $E_i = (H_i, T_i)$ of disjoint subsets of $\mathscr{V}$, where H_i denotes the vertices in the *head* and T_i the vertices in the *tail*. A path is a sequence of vertices and hyperedges $P_{v_i,v_{q+1}} = (v_i, E_i, ..., E_{j-1}, v_j, E_j, ..., E_q, v_{q+1})$ where $v_i \in T_i$, $v_{q+1} \in H_q$ and $v_j \in T_j \cap H_{j-1}$ for all $j = 2, ..., q$. A path is a cycle if v_{q+1} is in the tail of E_i [19].

A metabolic network $\mathcal{N}$ can be modeled as a hypergraph and is represented by the tuple $(\mathcal{M}, \mathcal{R}, \mathbf{S}, \mathbf{l}, \mathbf{u})$, where $\mathcal{M}$ is the set of internal metabolites and $\mathcal{R}$ is the set of reactions. The *stoichiometric matrix* $\mathbf{S} \in \mathbb{R}^{m \times n}$ captures the entire network, where m is the number of internal metabolites and n is the number of reactions, including internal reactions $\mathcal{I}$ and exchange reactions $\mathcal{E}$. Usually, the number of metabolites is much smaller than the number of reactions [36]. The columns of $\mathbf{S}$ correspond to the reactions in the network and the entries are the stoichiometric coefficients for each reaction and capture the mass balance for each metabolite. A negative coefficient indicates that a metabolite is consumed, called *reactant* and is in the tail of the hyperedge. Vice versa, if the coefficient is positive, the metabolite is produced, called *product*, and belongs to the head of the hyperedge.

Figure 7 shows a simplified metabolic network with three exchange reactions $\{r_1, r_4, r_8\}$ and five internal reactions $\{r_2, r_3, r_5, r_6\}$. The set of internal metabolites is $\{A, B, C, D, E, F\}$. The set of reversible reactions is $\{r_5, r_6, r_3\}$ and the set of irreversible reactions $\{r_1, r_2, r_4, r_7, r_8\}$. All irreversible reactions have to be greater or equal to zero. The metabolic network contains the cycle $(A, r_2, E, r_6, D, r_5, A)$.

Figure 7: simplified metabolic network



The corresponding stoichiometric matrix $\mathbf{S}$ is:

$$\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

$\mathbf{S}$ is a 6×8 matrix, where the rows correspond to the internal metabolites in alphabetical order, and the columns denote the reactions r_i in increasing order. In the exchange reaction r_1, A is produced, and therefore the corresponding column has a positive entry for A. In the internal reaction v_2, the consumption of A is captured by a -1 and the production of B and E are denoted by a 1 for each metabolite. In this example the quantities for the metabolites are either 0, 1 or -1 in all reactions. In realistic molecular models, the stoichiometric coefficients are always integer, but typically vary.

The matrix $\mathbf{S}_\mathcal{I}$ is the submatrix of $\mathbf{S}$ that contains the columns of internal reactions only. The stoichiometric matrix $\mathbf{S}$ relates the flux[1] vector $\mathbf{v}$ to the change in metabolite concentration $\mathbf{x}$: $\frac{d\mathbf{x}}{dt} = \mathbf{Sv}$ [28]. $\mathbf{l}$ and $\mathbf{u}$ capture the lower and upper bounds of the flux for all reactions. If the direction of a reaction is forced in one direction by the bounds, the reaction is said to be *irreversible*. Otherwise, it is *reversible*. The metabolic fluxes are given in the unit $= \frac{\text{mmol}}{\text{gDW}\times\text{h}}$, which denotes the flux of a metabolite normalized to the biomass of the cell.

[1] *Flux*, *reaction rate* and *flux distribution* are used interchangeably throughout the thesis.

3.2 Optimization for Metabolic Networks

In dynamic modelling, the dynamics of metabolite concentration are expressed by kinetic laws and kinetic parameters [36]:

$$\frac{dx_i}{dt} = f_i(x_1, x_2, ..., x_n; \Theta) \tag{30}$$

where Θ is the set of kinetic parameters and f_i models how the concentration of x_i changes depending on the concentration of the other metabolites. Such a dynamic model requires a lot of computational data in order to predict the metabolite concentration $\mathbf{x}$.

In constraint-based reconstruction and analysis (COBRA) methods, dynamic metabolic networks are modeled at *steady-state*:

$$\frac{d\mathbf{x}}{dt} = \mathbf{Sv} = \mathbf{0}$$

where $\mathbf{v} \in \mathbb{R}^n$ is the vector of flux distributions and v_i indicates the units of flow through reaction i. The stoichiometric matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ captures the topology of the molecular network. In COBRA methods, the set of possible flux distributions mathematically is analysed, whereas in dynamic modelling, the metabolite concentration is predicted. With COBRA methods one can predict the flux distribution under different environmental conditions and under changes in the environment [36]. Apart from the steady-state assumption, COBRA methods use context-specific constraints such as *reaction flux bounds* and *mass conservation*. The flux through a reaction is lower bounded by vector $\mathbf{l}$ and upper bounded by vector $\mathbf{u}$. Mass balance is captured in the stoichiometric matrix $\mathbf{S}$. As metabolic reactions take place fast compared to other reactions, such as cell division, the steady-state assumption is biologically plausible [40]. Compared to the dynamic modeling in Equation (30), COBRA methods require much less experimental data [36].

check with optimization section, mention dimension

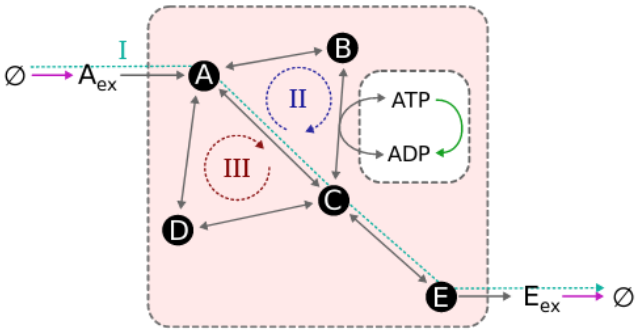
Extreme pathways are a set of vectors that satisfy the model constraints, such that any feasible flux can be written as a convex combination of extreme pathways [35]. There are three types of extreme pathways: *primary systemic pathways* (Type I), *futile cycles* (Type II) and *internal cycles* (Type III). Figure 8 shows the difference between the extreme pathways. Primary systemic pathways are pathways where exchange reactions are used[28]. These pathways are thermodynamically feasible and are desirable steady-state flux solutions (see Section 3.2.4). For most COBRA

methods, the goal is to compute a Type I pathway and forbidding Type II and Type III pathways.

In order to understand the difference between Type II and Type III pathways, we have to introduce *currency exchange reactions*. Currency exchange reactions are a subset of internal reactions, where the currency within the cell is changed [28]. For example, a reaction, where ATP is generated from ADP, transfers energy and is a Type II pathway.

Internal cycles are pathways where none of the exchange reactions and none of the currency exchange reactions are used and are thermodynamically infeasible. Futile cycles are pathways where none of the exchange reactions is used, but at least one currency exchange reaction is used [28]. *Energy reducing cycles* are futile cycles that are biologically feasible and should be allowed solutions, whereas *energy generating cycles* (EGCs) are futile cycles that are not wanted in a solution [28]. EGCs are problematic as they can lead to solutions that have a better objective value than when restricting the use of EGCs. One possibility to prevent EGCs is to force the directionality of the reaction.

Figure 8: Types of extreme pathways (taken from [28])



Primary systemic pathways (Type I) are biologically realistic, whereas internal cycles (Type III) are not thermodynamically feasible. Futile cycles (Type II) can either generate energy or reduce energy. Energy generating cycles are thermodynamically infeasible, whereas energy consuming cycles are realistic.

The feasible region of a COBRA model usually contains multiple solutions. We are interested in flux distributions that optimize a biological objective. A commonly used objective function is the maximization of growth. Growth can be incorporated into the stoichiometric matrix $\mathbf{S}$ by adding an extra column where the coefficients indicate the contribution to growth [31]. For an overview of biological objectives, see [32].

Depending on the constraints and the objective, we obtain a different COBRA variant. Each corresponds to an optimization problem which can be solved with

methods described in Section 2. Several COBRA methods relevant for this thesis are explained below.

3.2.1 Flux Balance Analysis

The most basic COBRA method is *flux balance analysis* (FBA). An optimal solution $\mathbf{v}^*$ to an FBA model, is a flux distribution maximizing a biological linear objective which respects the steady-state assumption and bound constraints.

FBA:

$$\max_{\mathbf{v}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{31a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{31b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{31c}$$

where $\mathbf{c} \in \mathbb{R}^n$ determines the linear objective function of the FBA. The structure of the network is captured in the stoichiometric matrix $\mathbf{S}$. The fluxes $\mathbf{v} \in \mathbb{R}^n$ are bounded by $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$. Constraint (31b) ensures the steady-state of the metabolite concentration. We are dealing with a linear program which can be solved efficiently (see Section 2.2).

As an example, let us consider the metabolic network in Figure 9. We have three internal metabolites A, B, C, two irreversible exchange reactions r_1, r_5 and three reversible internal reactions r_2, r_3, r_4. The stoichiometric matrix is:

$$\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix} \tag{32}$$

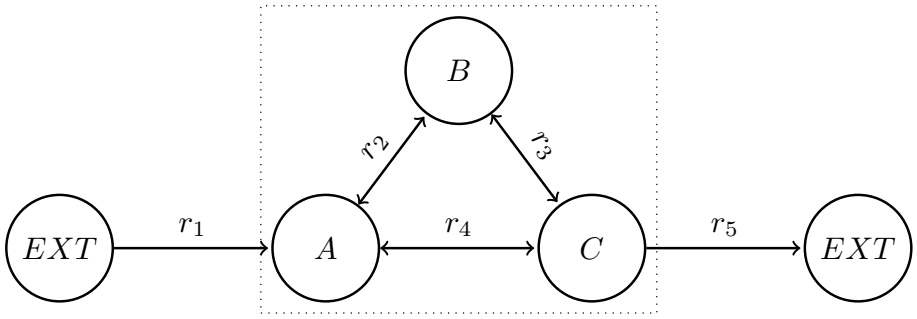
and we assume the following bounds on the fluxes:

$$\mathbf{l} = (0, -30, -30, -30, 0) \quad \mathbf{u} = (10, 30, 30, 30, 10) \tag{33}$$

If we maximize the flux through the internal reactions, that is $\mathbf{c} = (0, 1, 1, 1, 0)$, an optimal solution is $\mathbf{v}^* = (10, 30, 30, -20, 10)$. The solution contains an internal loop, as there is a flux of 20 going through the internal reactions.

A solution to FBA can be a convex combination of primary systematic pathways, futile cycles and internal cycles. To prevent biologically implausible futile cycles, the directionality is restricted with the bound constraints Problem (31) (c). Internal

Figure 9: simple model with internal loop



cycles are biologically not realistic, but are part of the solution space, posing a major drawback to FBA solutions.

3.2.2 CycleFreeFlux

Suppose we are given a solution $\mathbf{v}^{FBA}$ to an FBA problem (Problem (31)). The following linear program, known as *CycleFreeFlux* (CFF), returns a solution $\mathbf{v}$ that is structurally consistent with $\mathbf{v}^{FBA}$ and with the minimum sum of reaction rates such that the objective values are identical [15].

CFF:

$$\min_{\mathbf{v}} \quad \sum_i |v_i| = \sum_{i:v_i^{FBA}>0} v_i - \sum_{i:v_i^{FBA}<0} v_i \tag{34a}$$

$$\text{s.t.} \quad \mathbf{Sv} = \mathbf{0} \tag{34b}$$

$$\mathbf{c}^\intercal \mathbf{v} = \mathbf{c}^\intercal \mathbf{v}^{FBA} \tag{34c}$$

$$0 \leq v_i \leq v_i^{FBA} \quad \text{for } i \text{ with } v_i^{FBA} \geq 0 \tag{34d}$$

$$v_i^{FBA} \leq v_i \leq 0 \quad \text{for } i \text{ with } v_i^{FBA} < 0 \tag{34e}$$

$$v_i = v_i^{FBA} \quad \forall i \in \mathcal{E} \tag{34f}$$

Beside the steady-state assumption in Constraint (34b), the reaction direction has to match the direction in $\mathbf{v}^{FBA}$ or become zero, enforced by Constraint (34d) and Constraint (34c). In addition, the reaction rates of the exchange reactions $\mathcal{E}$ remain unchanged (Constraint (34f)). Constraint (34c) ensures that the objective value of $\mathbf{v}$ matches the objective value of $\mathbf{v}^{FBA}$. The objective function can be rewritten by replacing the absolute function by the sum of positive reaction rates minus the sum of negative reaction rates, as the sign of reaction rates is known from $\mathbf{v}^{FBA}$. As Problem (34) is a linear program, it is easy to solve (see Section 2.2.1). If the reactions contained in an internal loop are allowed to be zero, and if the objective does not depend on a reaction which is contained in an internal loop, $\mathbf{v}$ is thermodynamically

feasible [28].

other cases or disadvantages?

Let us go back to the FBA example visualized in Figure 9. If we apply CycleFreeFlux to the FBA solution $\mathbf{v}^{FBA} = (10, 30, 30, -20, 10)$ with $\mathbf{c}^\intercal \mathbf{v}^{FBA} = 40$, the solution flux $\mathbf{v}^* = (10, 30, 30, -20, 10)$ still contains an internal loop.

We will see in Section 3.2.4 that formulating a mathematical program such that the resulting flux is in the thermodynamically feasible subspace is $\mathcal{NP}$-hard.

explain enumeration of loops

3.2.3 Thermodynamic Flux Balance Analysis

The second law of thermodynamics states that the entropy in a closed system cannot decrease. In a metabolic network, each reaction i is associated with a scalar $\Delta\mu_i$ denoting the *Gibbs free energy change* or *potential difference*. The following has to hold [39]:

$$\Delta\mu_i = \Delta\mu_i^\circ + RT\ln(Q_i)$$

R is the universal gas constant, which relates energy to the amount of substance and temperature, and T the temperature in Kelvin. Q_i is the reaction quotient and denotes the ratio of product concentration to reactant concentration. $\Delta\mu_i^\circ$ is the *standard chemical potential*, or *equilibrium potential*, or *standard Gibbs free energy* of reaction i. Instead of using the reaction quotient, we can calculate the concentration of reactants and products in reaction i by using the stoichiometric coefficients $s_{i,*}$ and the metabolite concentration $\mathbf{c}$ [28]:

$$\Delta\mu_i = \Delta\mu_i^\circ + RT\sum_i s_{i,*}^\intercal \ln(\mathbf{c})$$

where $\Delta\mu_i$ is the change of Gibbs free energy for reaction i. Written in matrix notation, we have:

$$\boldsymbol{\Delta\mu} = \boldsymbol{\Delta\mu}^\circ + RTS^\intercal \ln(\mathbf{c}) \tag{35}$$

One method that integrates thermodynamic data to get solutions that are thermodynamically feasible is *thermodynamic flux balance analysis* (TFBA).

TFBA:

$$\max_{\mathbf{v},\mathbf{a},\mathbf{x},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{36a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{36b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{36c}$$

$$0 \leq M\mathbf{a}_i - \mathbf{v} \leq M \quad \forall i \in \mathcal{I} \tag{36d}$$

$$\epsilon \leq M\mathbf{a}_i + \boldsymbol{\Delta\mu} \leq M - \epsilon \quad \forall i \in \mathcal{I} \tag{36e}$$

$$\boldsymbol{\Delta\mu} = \boldsymbol{\Delta\mu}^\circ + RT * \mathbf{S}_\mathcal{I}^\intercal \mathbf{x} \tag{36f}$$

$$\ln(\mathbf{b}^L) \leq \mathbf{c} \leq \ln(\mathbf{b}^U) \tag{36g}$$

$$a_i \in \{0, 1\} \tag{36h}$$

The steady-state constraint (Constraint (36b)), the bound constraints (Constraint (36c)) and the linear objective function correspond to the FBA model (Problem (31)). Constraint (36d) and Constraint (36e) ensure that the Gibbs free energy is never reduced for any reaction i: $v_i = 0 \vee \text{sign}(v_i) = -\text{sign}(\Delta\mu_i)$. $M \in \mathbb{R}$ is a large constant that does not restrict the value of $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$. The binary variables $\mathbf{a}$ indicate the direction of $\mathbf{v}$: $a_i = 1$ implies $v_i > 0$ and $a_i = 0$ implies $v_i < 0$. If $a_i = 1$, v_i is zero or positive as $-M \leq -v_i \leq M$ holds. $\Delta\mu_i$ is not allowed to be zero so it is bounded by a small value ϵ. In that case, $\Delta\mu_i$ is negative as $\epsilon - M \leq \Delta\mu_i \leq -\epsilon$. If $a_i = 0$, $0 \leq -v_i \leq M$ holds and the flux v_i is zero or negative. $\Delta\mu_i$ is positive as $\epsilon \leq \Delta\mu_i \leq M - \epsilon$. $\mathbf{c}$ is the vector of metabolite concentrations which is bounded by $\ln(\mathbf{b}^L)$ and $\ln(\mathbf{b}^U)$. [28]

A solution to a TFBA model is thermodynamically feasible, however the model requires the standard equilibrium potential for each reaction $\boldsymbol{\Delta\mu}^\circ$ which is oftentimes not known. Additionally, the resulting problem is a mixed-integer problem and more difficult to solve than the FBA problem.

3.2.4 Loopless FBA

A simpler method to ensure that a solution does not contain an internal loop, which does not require additional thermodynamic data is *loopless FBA* (ll-FBA). To respect the second law of thermodynamics, the following inequality for reaction i is required:

$$\{v_i = 0\} \vee \{\text{sign}(v_i) = -\text{sign}(\Delta\mu_i)\} \tag{37}$$

which means that if a reaction carries flux, Gibbs free energy decreases [26]. A *loop* is a nonzero flux vector $\boldsymbol{\ell}$ such that the internal network is at steady-state: $\mathbf{S}_\mathcal{I}\boldsymbol{\ell} = \mathbf{0}$ [29]. As ℓ_i and $\Delta\mu_i$ have to be of opposite sign, unless $\ell_i = 0$, we know

that $\boldsymbol{\ell}^\intercal \boldsymbol{\Delta\mu} \neq \mathbf{0}$. However, *flux balance*, or *Kirchhoff's second law*, states that the reaction energies around a circuit have to sum up to zero, and therefore $\boldsymbol{\ell}$ is not thermodynamically feasible [39]. A flux distribution $\mathbf{v}$ contains a loop if there exists a nonzero vector $\boldsymbol{\ell}$ with $\mathbf{S}_\mathcal{I}\boldsymbol{\ell} = \mathbf{0}$ such that:

$$\text{sign}(\ell_i) \in \{\text{sign}(v_i), 0\} \quad \forall i \in \mathcal{I}$$

For a solution to be thermodynamically feasible, the reaction energies around any flux $\mathbf{v}$ have to sum up to zero: $\mathbf{v}^\intercal \boldsymbol{\Delta\mu} = 0$, where $\boldsymbol{\Delta\mu}$ is the vector of Gibbs free energy [39]. As we are interested in internal cycles, it is sufficient to verify that $\mathbf{v}_\mathcal{I}^\intercal \boldsymbol{\Delta\mu} = 0$, where $\mathbf{v}_\mathcal{I}$ is a flux distribution through internal reactions and $\boldsymbol{\Delta\mu}$ are the corresponding changes in Gibbs free energy. Any steady-state nonzero pathway that just uses internal reactions is a loop: $\mathbf{S}_\mathcal{I}\mathbf{v}_\mathcal{I} = \mathbf{0}$ [29]. The nullspace of $\mathbf{S}_\mathcal{I}$ is defined as $\text{null}(\mathbf{S}_\mathcal{I}) := \{\mathbf{x} \in \mathbb{R}^{|\mathcal{I}|} : \mathbf{S}_\mathcal{I}\mathbf{x} = \mathbf{0}\}$. Let $\mathbf{B} \in \mathbb{R}^{|\mathcal{I}| \times n}$ be a matrix with columns formed by the set of vectors $\{\mathbf{b}_i\}_{i=1}^n$ that form a basis for $\text{null}(\mathbf{S}_\mathcal{I})$. Any loop $\boldsymbol{\ell}$ is a linear combination of the nullspace of $\mathbf{S}_\mathcal{I}$ and can be written as $\boldsymbol{\ell} = \mathbf{B}^\intercal\boldsymbol{\alpha}$, where coefficients $\alpha_i \in \mathbb{R}$ [39]. Therefore, the following removes solutions with internal loops: $\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0}$. Adding the loopless constraints to the FBA problem, we obtain the following mathematical program:

$$\max_{\mathbf{v}, \boldsymbol{\Delta\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{38a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{38b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{38c}$$

$$\Delta\mu_i v_i < 0 \vee v_i = 0 \quad \forall i \in \mathcal{I} \tag{38d}$$

$$\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0} \tag{38e}$$

We can write Constraint (38e) as $\boldsymbol{\Delta\mu} = \mathbf{S}^\intercal\boldsymbol{\mu}$ [29, 39, 26]. Let us define the vector of potential differences of the internal reactions as $\boldsymbol{\Delta\mu} = \mathbf{S}^\intercal\boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is approximately the chemical potential. Looking only at internal reactions, $\boldsymbol{\Delta\mu}$ is defined as $\boldsymbol{\Delta\mu} = \mathbf{S}_\mathcal{I}^\intercal\boldsymbol{\mu}$. Each vector in the rowspace of $\mathbf{S}_\mathcal{I}$ is orthogonal to each vector in the nullspace of $\text{null}(\mathbf{S}_\mathcal{I})$ [29]. Therefore, $\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0}$.

read paper again

Rewriting Constraint (38e), we obtain the following mathematical program [26]:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{39a}$$

$$\text{s.t.} \quad \mathbf{Sv} = \mathbf{0} \tag{39b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{39c}$$

$$\Delta\mu_i v_i < 0 \vee v_i = 0 \quad \forall i \in \mathcal{I} \tag{39d}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_\mathcal{I}^\intercal\boldsymbol{\mu} \tag{39e}$$

where $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\Delta\mu} \in \mathbb{R}^{|(\mathcal{I})|}$. Problem (39) is $\mathcal{NP}$-hard and much more complicated to solve than FBA [13]. We use $\boldsymbol{\Delta\mu}$ and $\boldsymbol{\mu}$ due to the biological interpretability, however it is not necessary to define $\boldsymbol{\Delta\mu}$ as in only appears in Constraint (39e).

Constraint (d) in Problem (39) and Problem (38) poses a challenge due to the disjunction, the strict equality and the product of decision variables $\boldsymbol{\Delta\mu}$ and $\mathbf{v}$. To simplify the model, the disjunction is rewritten and we arrive at the loopless FBA model used in this thesis.

ll-FBA:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{40a}$$

$$\text{s.t.} \quad \mathbf{Sv} = \mathbf{0} \tag{40b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{40c}$$

$$((v_i \geq 0) \wedge (\Delta\mu_i \leq -\epsilon)) \vee ((v_i \leq 0) \wedge (\Delta\mu_i \geq \epsilon)) \quad \forall i \in \mathcal{I} \tag{40d}$$

$$\boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal\mathbf{S}_\mathcal{I} \tag{40e}$$

Whereas the value of $\boldsymbol{\Delta\mu}$ in TFBA corresponds to the Gibbs free energy change tested in experiments, in ll-FBA only the sign($\boldsymbol{\Delta\mu}$) corresponds to the actual Gibbs free energy change [39]. As the value does not matter, we can scale $\boldsymbol{\Delta\mu}$ by not allowing it to be in the interval $[-\epsilon, \epsilon]$. The resulting problem is a disjunctive program with linear constraints in the disjunctions and can be reformulated as a mixed-integer program with indicator or big-M constraints (see Section 2.4).

Going back to the example in Figure 9 with S defined in Equation (32) and flux bounds defined in Section 4.4, the loopless flux distribution is $\mathbf{v}^* = (10, 10, 10, 0, 10)$ with $\mathbf{c}^\intercal\mathbf{v}^* = 20$.

Loopless FBA excludes internal cycles from the solution space. However, energy

generating cycles still have to be blocked by restricting the directionality.

3.2.5 st-FBA

Semi-thermodynamic Flux Balance Analysis (st-FBA) combines the loopless FBA and TFBA approaches [28]. The loopless FBA formulation is extended by constraining the Gibbs free energy change $\boldsymbol{\Delta\mu}$ of a subset of reactions. Metabolites that carry energy such as ATP and their related compounds such as ADP and AMP are treated as in TFBA. The chemical potential of such a metabolite is limited by bounds known from experiments. Bounding the chemical potential of these energy-carrying currency metabolites excludes energy generating cycles, and it is no longer required to force the direction of certain reactions as in ll-FBA. st-FBA is thus a trade off between the $\boldsymbol{\Delta\mu}$ values corresponding to the Gibbs free energy changes which requires experimental data, and the approximation of the Gibbs Free energy changes which only deals with the directionality. We require more experimental data than for ll-FBA, as we need the bounds of reactions involved in energy reducing cycles. As we add the experimental data just on a small subset of reactions, just a fraction of information needed for TFBA is needed for st-FBA.

3.2.6 Enzyme Constrained Metabolic Models

In FBA, the optimal reaction rate of a metabolic network is limited by the nutrition uptake. However, the flux depends also on the enzyme abundances catalysing the reactions. Especially modelling physiological responses, such as *overflow metabolism*, requires enzymatic data [38]. Overflow metabolism refers to the phenomenon that a cell uses fermentation instead of respiration, even though respiration is more energetically efficient [5]. There are several hypotheses to explain the metabolic switch from respiration to fermentation. One hypothesis is that overflow metabolism is connected to the enzyme mass. When using fermentation, the cell does not grow as fast as with respiration, but it yields more energy with the same enzyme mass [38]. *GECKO* is one method that integrates enzyme constraints into a genome-scale model. The following section is based on [38].

Suppose reaction j is catalysed by enzyme E_i. The *turnover number* indicates how efficient an enzyme is. The reaction rate v_j depends on the *enzyme usage* e_i and the turnover number k_{cat}^{ij}:

$$k_{cat}^{ij} * e_i = v_j \tag{41}$$

The turnover number is given in $\frac{1}{\mathrm{h}}$, the enzyme usage in $\frac{\mathrm{mmol}}{\mathrm{gDW}}$, such that $\mathbf{v}$ has the unit of $\frac{\mathrm{mmol}}{\mathrm{gDW} \times \mathrm{h}}$. Equation (41) is also known as *enzyme mass balance*. To account for

enzyme mass balance in a GEM, matrix S^{GECKO} is constructed by extending the stoichiometric matrix $\mathbf{S}$ by three submatrices. A row is added for each enzyme E_i and a column for the corresponding enzyme usage e_i, with p enzymes. The lower left submatrix contains the enzyme information on the diagonal, that is $-1/k_{cat}^{ij}$. The lower right matrix is the identity matrix. The upper right matrix contains only zeros. The resulting matrix is of the form:

$$S^{GECKO} = \left[\begin{array}{ccc|ccc} s_{1,1} & \dots & s_{1,n} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & \dots & s_{m,n} & 0 & \dots & 0 \\ \hline -1/k_{cat}^{11} & \dots & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -1/k_{cat}^{pn} & 0 & \dots & 1 \end{array}\right] = \left[\begin{array}{cc} \mathbf{S} & \mathbf{0}_{m,p} \\ \text{diag}(-1/k_{cat}^{ij}) & \mathbf{I}_p \end{array}\right]$$

In addition to the reaction rates $\mathbf{v}$, we are interested in the enzyme usage $\mathbf{e}$. We obtain the following optimization problem:

$$\max_{\mathbf{v},\mathbf{e}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{42a}$$

$$\text{s.t.} \quad S^{GECKO}(\mathbf{v}, \mathbf{e}) = \mathbf{0} \tag{42b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{42c}$$

$$\mathbf{0} \leq \mathbf{e} \leq [\mathbf{E}] \tag{42d}$$

The objective function and Constraint (42c) are also part of the FBA problem. Constraint (42b) forces a steady-state of the metabolites and enzyme mass balance. As the upper right matrix contains only zeros, the steady-state constraint $\mathbf{Sv} = \mathbf{0}$ is preserved. In addition to bounds on the fluxes, Constraint (42d) limits the enzyme usage e_i, which cannot be negative and has to be below the intracellular enzyme concentration $[E_i]$:

$$0 \leq e_i \leq [E_i]$$

Together both lower submatrices of S^{GECKO} make up the enzyme mass balance constraints:

$$\begin{aligned} -\frac{1}{k_{cat}^{ij}} v_j + e_i &= 0 \\ e_i &= \frac{1}{k_{cat}^{ij}} v_j \\ k_{cat}^{ij} * e_i &= v_j \end{aligned}$$

Depending on the type of enzyme, S^{GECKO} is modified slightly. If different enzymes catalyze the same reaction, they are called *isozymes* and one mass balance constraints for each isozyme is added to the model.

proteins or enzymes in gecko model? compare to methods gecko models

If the intracellular enzyme concentration is not known, the total enzyme abundance is limited. Constraint (42d) is replaced by:

$$\sum_{i}^{p} \mathrm{MW}_i e_i \leq P$$

where MW_i is the molecular weight of enzyme i and P is the total protein content in the cell [38].

4 Methods

differentiate between cycle in network and loop in flux

mention scaling of epsilon; why eps = 1 in computation

In Section 3, we have seen how mathematical optimization can be used to predict fluxes $\mathbf{v}$ in a cell that optimize a biological objective. In particular ll-FBA (Problem (40)) can be used to predict fluxes that do not contain internal loops. However, ll-FBA is difficult to solve due to the constraint that $\Delta\mu_i$ and v_i have to be of opposite sign, unless $v_i = 0$, which is modeled by a disjunction. In this thesis, we compare the runtime of the big-M reformulation of ll-FBA to the convex-hull reformulation and to solving the problem by adding cutting planes.

technical setup, github link

The code is written in Julia 1.9.0. We use JuMP [25] and MathOptInterface [23] to build the mathematical models. The MIP solver used in the experiments is SCIP [6]. The LP solver used is HiGHS [20]. We use COBREXA[11] to load the biological model data.

rewrite following section, shorten section

present more why you have done what and then link the sections

In Section 4.1, we explain briefly how we test for a solution whether it does not contain internal loops. In Section 4.2, we see how ll-FBA can be reformulated with indicator constraints, and with big-M constraints. In Section 4.3, we take a solution to the FBA problem (Problem (31)) which may contain internal loops. Loops are identified and cuts are added to the ll-FBA formulation to exclude solutions that contain a specific loop from the feasible region. In Section 4.4, we solve a relaxed version of ll-FBA, where we solve an FBA, but assign binary variables dependent on the directionality of $\mathbf{v}$. If a solution to the relaxed problem is not a feasible solution to the ll-FBA problem, a cut is added to the relaxed problem that cuts off the assignment of binary variables. This procedure is repeated until a solution is found that is a feasible solution to ll-FBA. We will see in Section 4.5 how to generate stronger cuts than a no-good cut using combinatorial Benders' cuts. The ll-FBA problem is split into a master problem and subproblem and we exploit the duality of the infeasible subproblem to generate cuts that include a subset of binary variables only. Another type of cuts are intersection cuts, explained in Section 4.6. The feasible region is split into the polyhedral set P, which contains the inequality and equality constraints, and the set S, which contains the disjunctions. An S-free

set C is defined around a solution $\tilde{\boldsymbol{x}}$ that lies in P but not in S. The intersection cut is the hyperplane going through the intersection of the conic relaxation at $\tilde{\boldsymbol{x}}$ with C. In Section 4.7, the ll-FBA problem is written as disjunctive problem, which includes a binary variable for each disjunct. The problem is then solved by using the big-M reformulation and the convex-hull reformulation. In Section 4.8 we introduce the data that is used for the experiments.

4.1 Feasibility Test

For a given flux distribution $\tilde{\boldsymbol{v}}$ and the corresponding directions $\tilde{\boldsymbol{a}}$ the following problem is feasible if it is loopless:

$$\max_{\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad 0 \tag{43a}$$

$$\text{s.t.} \quad -M \leq \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} : \tilde{a}_i = 1 \tag{43b}$$

$$\epsilon \leq \Delta\mu_i \leq M \quad \forall i \in \mathcal{I} : \tilde{a}_i = 0 \tag{43c}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^{\intercal}\boldsymbol{\mu} \tag{43d}$$

where we set the big-M constant to 1000 and ϵ to 1.

Note that if $\tilde{v}_i = 0$, the reaction can be removed from the problem by removing the reaction from $\mathbf{S}_{\mathcal{I}}$ and without adding variable $\Delta\mu_i$. Problem (43) can also be used to test the looplessness of a solution and a subset of internal reactions by setting the flux of all other reactions to zero.

Going back to the example visualized in Figure 9 in Section 3.2.1, for the solution $\tilde{\boldsymbol{v}} = (10, 30, 30, -20, 10)$ where $\tilde{v}_2, \tilde{v}_3, \tilde{v}_4$ are the fluxes through the internal reactions, and $\tilde{\boldsymbol{a}} = (1, 1, 0)$. The solution contains a loop and Problem (43) is infeasible.

For the loopless solution $\tilde{\boldsymbol{v}} = (10, 10, 10, 0, 10)$, we only look at the non-zero internal fluxes $\tilde{v}_2, \tilde{v}_3$ and the corresponding directionality is $\tilde{\boldsymbol{a}} = (1, 1)$. Problem (43) is feasible, and the solution is $\boldsymbol{\Delta\mu} = (-1, -1)$ and $\boldsymbol{\mu} = (2, 1, 0)$.

4.2 Loopless FBA Formulations

We can reformulate Problem (40) to no longer write the disjunctions explicitly. The mixed-integer program of flux balance analysis without unbounded internal cycles using indicator constraints [39]:

ll-FBA (indicator):

$$\max_{\mathbf{v},\mathbf{a},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{44a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{44b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{44c}$$

$$a_i = 1 \implies v_i \geq 0 \quad \forall i \in \mathcal{I} \tag{44d}$$

$$a_i = 1 \implies \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} \tag{44e}$$

$$a_i = 0 \implies v_i \leq 0 \quad \forall i \in \mathcal{I} \tag{44f}$$

$$a_i = 0 \implies \Delta\mu_i \geq \epsilon \quad \forall i \in \mathcal{I} \tag{44g}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^\intercal \boldsymbol{\mu} \tag{44h}$$

The mixed-integer program of flux balance analysis without unbounded internal cycles using big-M constraints [39]:

ll-FBA (big-M):

$$\max_{\mathbf{v},\mathbf{a},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{45a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{45b}$$

$$l \leq v \leq u \tag{45c}$$

$$-Ma_i + \epsilon(1 - a_i) \leq \Delta\mu_i \leq -\epsilon a_i + M(1 - a_i) \quad \forall i \in \mathcal{I} \tag{45d}$$

$$-M(1 - a_i) \leq v_i \leq Ma_i \quad \forall i \in \mathcal{I} \tag{45e}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^\intercal \boldsymbol{\mu} \tag{45f}$$

The big-M formulation ensures the opposite sign of v_i and $\Delta\mu_i$ if $v_i \neq 0$. If the flux through reaction i is a forward flux, that is $v_i > 0$ and $a_i = 1$, it holds that $-M \leq \Delta\mu_i \leq -\epsilon$ and analogously for a backward flux. We set ϵ to 1, and the big-M constant is the maximal absolute value of the flux bounds $\mathbf{l}, \mathbf{u}$.

reference scaling again

As explained in Section 3.2.4, Constraints (44h) and (45f) can be replaced by $\mathbf{B}^\intercal \boldsymbol{\Delta\mu} = \mathbf{0}$.

4.3 Blocking Cycles

use CycleFreeFlux instead?

mention CycleFreeFVA here or in bio section

We take an optimal solution $\mathbf{v}^*$ to Problem (31). The metabolic network and the $\mathbf{v}^*$ are transformed such that the hyperarcs are split into simple directed edges. We search for loops in the transformed graph on edges with nonzero flux. Finally, we check whether the loop found in the transformed graph is a thermodynamically infeasible loop in the metabolic network. An internal loop can be removed by adding a constraint to Problem (39).

The hyperarcs represented by $\mathbf{S}$ are split up by replacing each hyperarc $E = (H, T)$ by k simple directed edges for every pair in $H \times T$. The transformed matrix is denoted by $\bar{\mathbf{S}}$. Assume the flux though hyperarc E is v_E^*. $\bar{\mathbf{v}}$ has then n entries with v_E^*. For every nonzero reaction in v^*, the corresponding reactions of $\bar{\mathbf{S}}$ are used to construct a simple directed graph G. We use the Graph package to build the graph and simplecycles_iter to compute the set of cycles in G [18]. We test for a cycle in the transformed network, whether the corresponding reactions in $\mathbf{S}$ are thermodynamically feasible (see Section 4.1). To block a given cycle, a Boolean expression is added as a constraint ensuring that the a_i's for reactions i in the cycle are assigned differently.

Suppose we have a solution that contains a loop $\boldsymbol{\ell}$ and let $\mathbf{a}$ be the corresponding binary variables indicating the direction of the flux. We can write the assignment of binary variables as Boolean expression: $t_1 \wedge ... \wedge t_i \wedge ...t_{|\mathcal{I}|}$, where $t_i = a_i$ if $a_i = 1$ and otherwise $t_i = 1 - a_i$. To block the cycle, the assignment of the a_i's has to be constrained: $\neg(t_1 \wedge ... \wedge t_i \wedge ...t_{|\mathcal{I}|}) = 1$. Formulating the expression as linear constraint yields: $t_1 + ... + t_i + ...t_{|\mathcal{I}|} \geq 1$.

A detected cycle can be blocked by a cut in the thermodynamic feasible FBA. The solution will not be impacted, but the run time can be affected. We filter the thermodynamically infeasible cycles in the set of cycles found in the transformed graph. Instead of blocking all detected cycles, we experiment with blocking the smallest cycles, which potentially leads to stronger cuts.

example

4.4 No-Good Cuts

When using no-good cuts to solve ll-FBA, the solving procedure is divided into first solving the relaxed problem and then checking the feasibility of the solution to the relaxed problem. The following relaxed version of Problem (44) is solved, where the

loopless constraints are ignored, but the indicator variables $\mathbf{a}$ are assigned:

$$\max_{\mathbf{v},\mathbf{a}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{46a}$$

$$\text{s.t.} \qquad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{46b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{46c}$$

$$a_i = 1 \quad \Longrightarrow \quad v_i \geq 0 \quad \forall i \in \mathcal{I} \tag{46d}$$

$$a_i = 0 \quad \Longrightarrow \quad v_i \leq 0 \quad \forall i \in \mathcal{I} \tag{46e}$$

The indicator constraints can be linearized by using the big-M method. If a solution $\mathbf{x}^{IP}$ to Problem (46) is thermodynamically infeasible, the following constraint is added:

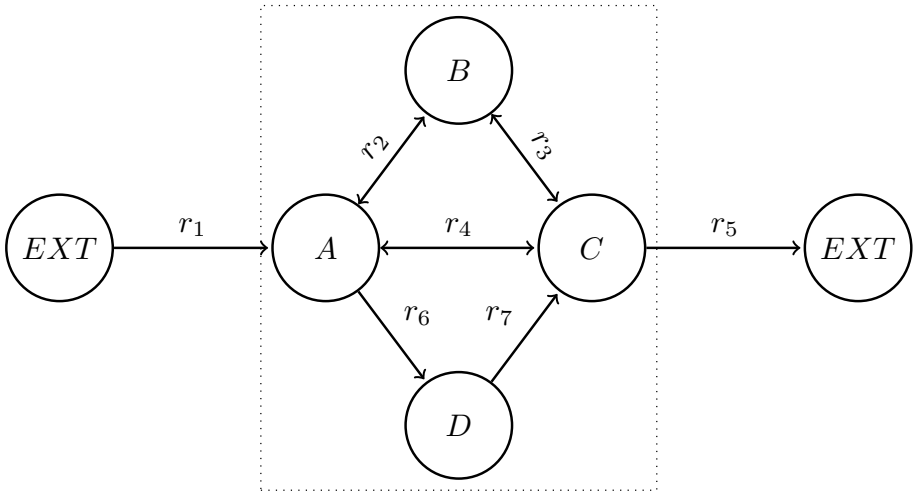
$$\sum_{i \in \mathcal{I}: x_i^{IP}=1} a_i + \sum_{i \in \mathcal{I}: x_i^{IP}=0} (1 - a_i) \leq |\mathcal{I}| - 1 \tag{47}$$

The problem is now solved again and this procedure is repeated until a solution is thermodynamically feasible.

As an example, let us consider the metabolic network in Figure 10. We have four internal metabolites A, B, C, D, two irreversible exchange reactions r_1, r_5, two irreversible internal reactions r_6, r_7 and three reversible internal reactions r_2, r_3, r_4. We assume the following bounds on the fluxes:

$$\mathbf{l} = (0, -10, -10, -20, 0, 0, 0) \quad \mathbf{u} = (20, 30, 30, 30, 20, 10, 10)$$

Figure 10: simple model with two internal loops



show flux in color

If we maximize the flux through the all reactions, that is $\mathbf{c} = \mathbf{1}$, the optimal solution is $\mathbf{v}^* = (20, 30, 30, -20, 20, 10, 10)$ and $\mathbf{a} = (1, 1, 0, 1, 1)$. The solution is not thermodynamically feasible as it contains two internal loops. The corresponding no-good cut is:

$$(1 - a_3) + \sum_{i \in \{1,2,4,5\}} a_i \leq 5 - 1$$

4.5 Combinatorial Benders' Cuts

When deriving combinatorial Benders' cuts for ll-FBA, the solving procedure is divided into first solving the relaxed problem and then checking the feasibility of the solution to the relaxed problem, similar to the no-good cut derivation. However, instead of forbidding the infeasible solution found to the relaxed problem, we aim at identifying the variables that lead to the thermodynamic infeasibility, and potentially derive stronger cuts.

ll-FBA matches the required problem form Problem (16). We have a linear objective $\mathbf{c}^\intercal \mathbf{v}$, continuous variables $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$, and a set of binary variables $\mathbf{a}$. There are linear constraints on the continuous variables Constraints (44b), (44c) and (44h), and no constraints on the binary variables. The continuous variables and the binary variables are only connected by the indicator constraints (Constraints (44d–g)), and we can split the MIP into a master and a subproblem. The master problem (MP) is Problem (46) and the subproblem (SP) is the following *parameterised program*:

$$\begin{aligned}
\max_{\boldsymbol{\mu}, \boldsymbol{\Delta\mu}} \quad & 0 && && && (48a)\\
\text{s.t.} \quad & a_i^{MP} = 1 && \implies && \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} && (48b)\\
& a_i^{MP} = 0 && \implies && \Delta\mu_i \geq \epsilon \quad \forall i \in \mathcal{I} && (48c)\\
& && \boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^\intercal \boldsymbol{\mu} && && (48d)
\end{aligned}$$

where $\mathbf{a}^{MP}$ are the values of a solution to the master problem. If the subproblem is infeasible, we compute a corresponding minimal infeasible subsystem (MIS) $\mathcal{C}$. The minimal infeasible subsystem contains a minimal number of constraint indices that make the subproblem infeasible. The following combinatorial Benders' (CB) cut is added to the master problem if the subproblem is infeasible:

$$\sum_{i \in \mathcal{C}: a_i^{MP} = 0} a_i + \sum_{i \in \mathcal{C}: a_i^{MP} = 1} (1 - a_i) \geq 1$$

Note that if $\mathcal{C}$ contains all internal reactions $\mathcal{I}$, the CB cut corresponds to a no-good cut (Equation (47)).

The pseudocode for the combinatorial Benders' cut procedure is shown in Algorithm 1. In build_master_problem, the MP is built, which is the FBA problem with binary variables $\mathbf{a}$ that are linked to the directionality of $\mathbf{v}$. The relation between the flux variables $\mathbf{v}$ and the binary variables $\mathbf{a}$ can be expressed by indicator constraints as in Problem (46), by big-M constraints, or by indicator constraints and big-M constraints simultaneously. The big-M constant corresponds to the maximal absolute value of lower bounds $\mathbf{l}$ and upper bounds $\mathbf{u}$. The set of minimal infeasible subsystems is computed in compute_mis, which identifies at most m subsystems. The MIS search is explained in detail in Section 4.5. The dual problem of the infeasible subproblem is built with the Dualization package [16]. If we find a minimal infeasible subset $\mathcal{C}$, a combinatorial Benders' cut is added to the master problem. If there exists no MIS, the subproblem is feasible and thus the solution to the master problem is thermodynamically feasible. This process is repeated until a solution is found, that is feasible in the master problem and the subproblem and therefore also for Problem (44). In build_sub_problem, the subproblem is built based on the thermodynamically feasible solution of the master problem to obtain the corresponding values for $\boldsymbol{\Delta\mu}$ and $\boldsymbol{\mu}$.

Algorithm 1 ll-FBA with combinatorial Benders' cuts

Require: stoichiometric matrix $\mathbf{S}$, lower bound on fluxes $\mathbf{l}$, upper bound on fluxes $\mathbf{u}$, allowed number of cuts per iteration m, indices of internal reactions $\mathcal{I}$

```
MP ← build_master_problem(S, l, u, I)
v, a ← optimize(MP)
C ← compute_mis(S_I, a, m)              ▷ set of minimal infeasible subsystems
while C ≠ ∅ do
    add_cut(MP, a, C)                   ▷ combinatorial Benders' cut
    v, a ← optimize(MP)
    C ← compute_mis(S_I, a, m)
end while
SP ← build_sub_problem(S_I, I, a)
Δμ, μ ← optimize(SP)
return (v, a, Δμ, μ)                    ▷ loopless flux distribution
```

Going back to the metabolic network in Figure 10 with the objective function $f = \max \mathbf{1}^\intercal \mathbf{v}$, the optimal solution of the master problem is $\mathbf{v}^* = (20, 30, 30, -20, 20, 10, 10)$ and $\mathbf{a} = (1, 1, 0, 1, 1)$. The solution is not thermodynamically feasible, and a minimal infeasible subset is $\mathcal{C} = \{3, 4, 5\}$, which corresponds to the cycle $r_4 = -10, r_6 =$

$10, r_7 = 10$. The corresponding combinatorial Benders' cut is:

$$a_3 + 1 - a_4 + 1 - a_5 \geq 1$$

MIS Search

If a solution to the master problem is not loopless, the subproblem is infeasible. We use the infeasible subproblem and its corresponding dual problem to generate a minimal infeasible subsystem.

First, the primal problem (Problem (48)) is rewritten in inequality form. To have the decision variable $\boldsymbol{\mu}$ on the left-hand side of $\leq$, the Constraint (48c) is multiplied by -1:

$$\Delta\mu_i \geq \epsilon \implies -\Delta\mu_i \leq -\epsilon$$

We aslo subsitute $\boldsymbol{\Delta\mu}$ with $\mathbf{S}_{\mathcal{I}}^{\intercal}\boldsymbol{\mu}$ and obtain:

$$\max_{\mathbf{x}} \quad 0 \tag{49a}$$

$$\text{s.t.} \quad \tilde{\boldsymbol{A}}\boldsymbol{\mu} \leq \tilde{\boldsymbol{b}} \tag{49b}$$

where

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} [\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 1 \\ -[\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 0 \end{bmatrix} \quad \tilde{\boldsymbol{b}} = \begin{bmatrix} -\epsilon^{|\mathcal{C}|} \end{bmatrix}$$

We compute minimal infeasible subsystems by modifying the dual problem as explained in Section 2.5.2 (see Problem (21)). The objective function of the dual is moved to the constraint $-\tilde{b}^{\intercal}\lambda = -1$, and we minimize $\sum_i w_i\lambda_i$. By modifying $\mathbf{w}$, we potentially derive several minimal infeasible subsystems to one infeasible solution. Each solution corresponds to a minimal infeasible subsystem. The nonzero elements in λ correspond to a MIS with the set of reaction indices $\mathcal{C}$. We choose how many cuts k can be added per iteration relative to the model size. For any $i \in \{1, ..., k\}$, we set the i-th coefficient in the objective function to zero and set all other coefficients to 1. At the end of the MIS search, we filter the minimal infeasible subsystems found to have unique subsystems within each iteration.

should we not minimise over the number of nonzero lambdas?

4.6 Intersection Cuts

reference optimization section more

In order to apply intersection cuts to the loopless FBA problem Problem (40), the problem has to be brought in the required form:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in S \cap P \end{aligned}$$

We define $\mathbf{x} = (\mathbf{v}, \boldsymbol{\Delta\mu}, \boldsymbol{\mu})$, where $\mathbf{v} \in \mathbb{R}^n$, $\boldsymbol{\Delta\mu} \in \mathbb{R}^{\dim(\mathcal{I})}$ and $\boldsymbol{\mu} \in \mathbb{R}^m$. The vector $\mathbf{c} \in \mathbb{R}^n$ has the same coefficients as the objective function in FBA Problem (31) and a coefficient of 0 for the variables $\boldsymbol{\Delta\mu}, \boldsymbol{\mu}$. The polyhedral set P contains solutions that respect the steady-state constraints, the bound constraints on $\mathbf{v}$ and Constraint (40e):

$$P = \{(\mathbf{v}, \boldsymbol{\Delta\mu}, \boldsymbol{\mu}) \,|\, \mathbf{Sv} = \mathbf{0},\, \mathbf{l} \leq \mathbf{v} \leq \mathbf{u},\, \boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal \mathbf{S}_\mathcal{I}\}$$

Constraint (40d) is captured in the set S by using the same approximation as in Problem (44) and Problem (45): for any internal reaction i the flux through the reaction v_i and $\Delta\mu_i$ are of opposite sign unless $v_i = 0$ in which case $\Delta\mu_i$ can be positive or negative but cannot be in the interval $(-\epsilon, \epsilon)$. S is closed as it is defined using inclusive inequalities, and therefore it contains its limit points.

$$S = \{(v_i, \Delta\mu_i) \,|\, (v_i \geq 0,\, \Delta\mu_i \leq -\epsilon) \vee (v_i \leq 0,\, \Delta\mu_i \geq \epsilon) \quad \forall i \in \mathcal{I}\}$$

The intersection of P and S is the feasible region of the ll-FBA problem.

define A, b (to understand what $\tilde{A}$) means later

As explained in Section 2.5.3, in order to get an intersection cut we require an S-free set C containing the relaxed solution $\tilde{\boldsymbol{x}}$. Let $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\Delta\mu}}, \tilde{\boldsymbol{\mu}})$ be an optimal solution to the relaxed LP which is in P but not in S. There must be at least one reaction i for which holds $\Delta\mu_i v_i > 0$

include interval -1, 1

. One of the maximal S-free sets below contains such an optimal solution:

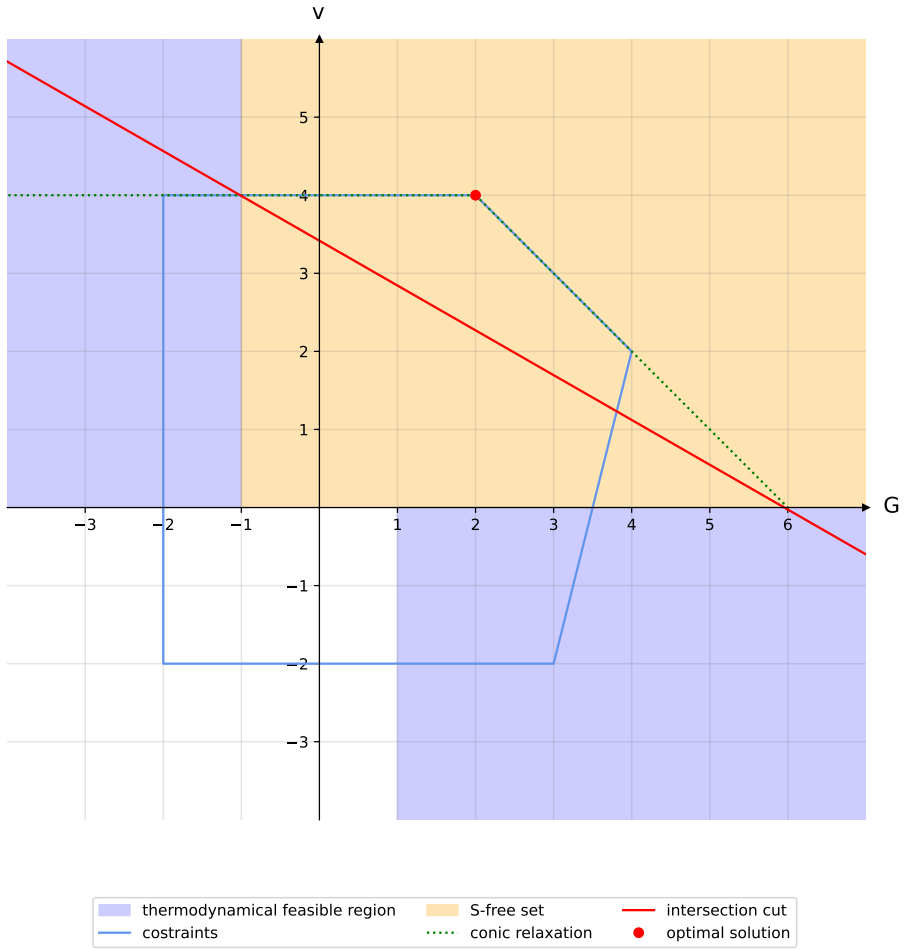
$$\begin{aligned} C_1 &= \{(v_i, \Delta\mu_i) \,|\, \Delta\mu_i \geq -\epsilon,\, v_i \geq 0\} \\ C_2 &= \{(v_i, \Delta\mu_i) \,|\, \Delta\mu_i \leq \epsilon,\, v_i \leq 0\} \end{aligned}$$

Suppose that $\tilde{v}_i > 0$ and $\tilde{\Delta\mu}_i > -\epsilon$. Such a solution is not in S but contained in C_1

projected into space of $(v_i \Delta\mu_i)$

. A diagram for this case is shown in Figure 6. If $\tilde{v}_i < 0$ and $\tilde{\Delta\mu}_i < \epsilon$ the solution is contained in C_2.

Figure 11: diagram of an intersection cut for ll-FBA



We consider this simple model with constraints in 2D, with $\epsilon = 1$. We have the decision variables $v_i, \Delta\mu_i \in \mathbb{R}$, polyhedral constraints P (blue), and the set S (purple area). The intersection of S and P is the true feasible region. Let $(\tilde{\Delta\mu}_i, \tilde{v}_i)$ of the optimal solution to the relaxed problem be at $(2, 4)$. As $\tilde{\Delta\mu}_i$ and $\tilde{v}_i$ are both positive, C_1 is a maximal S-free set (yellow area). The conic relaxation at $(\tilde{\Delta\mu}_i, \tilde{v}_i)$ and the S-free set C_1 intersect at the points $(-1, 4)$ and $(6, 0)$. The area between the line through the intersecting points and $(\tilde{\Delta\mu}_i, \tilde{v}_i)$ is cut off.

make figure nice, correct labels, colour interior of polytope, fix cut off extreme ray, correct subsection

The conic relaxation P' at a relaxed solution $\tilde{\boldsymbol{x}}$ is a cone with apex $\tilde{\boldsymbol{x}}$ the extreme rays are the intersections of the active constraints. P' can be written as a conic combination of extreme rays Equation (23) or as a polyhedron of the active constraints

Equation (24). The basic variables and nonbasic constraints defining a basis P' for $\tilde{\boldsymbol{x}}$ can be extracted from an LP solver.

Suppose reaction i in $\tilde{\boldsymbol{x}}$ has $\tilde{v}_i > 0$ and $\tilde{\Delta\mu}_i > -\epsilon$. $(\tilde{v}_i, \tilde{\Delta\mu}_i) \in C_1$ and $(\tilde{v}_i, \tilde{\Delta\mu}_i) \notin C_2$. To compute the intersection of the conic relaxation and C_1, we check for each extreme ray $\mathbf{r}^j = (-\tilde{\mathbf{A}}^{-1}_{*,j})$ generated by $\tilde{x}$ whether it intersects the boundary of C_1. First, the intersection of the conic relaxation with the hyperplane $h_1 = \{\mathbf{x} \in \mathbb{R}^{n+m+|\mathcal{I}|} \mid v_i = 0\}$ is computed. $(\tilde{\boldsymbol{x}} + \lambda_1 \mathbf{r}^j)_{k_1} = \mathbf{0}$ is solved for λ_1, where k_1 corresponds to the index of v_i in $\tilde{\boldsymbol{x}}$.

Similarly, the intersection of each extreme ray generated by $\tilde{\boldsymbol{x}}$ with h_2 is computed, where $h_2 = \{\mathbf{x} \in \mathbb{R}^{n+m+|\mathcal{I}|} \mid \Delta\mu_i = -\epsilon\}$. We solve $(\tilde{\boldsymbol{x}} + \lambda_2 \mathbf{r}^j)_{k_2} = -\boldsymbol{\epsilon}$ for the step size λ_2, where k_2 is the index of $\Delta\mu_i$ in $\tilde{\boldsymbol{x}}$. An extreme ray intersects the boundary of C_1 if either λ_1 or λ_2 is finite and positive. If λ_1, λ_2 are both negative or if the ray does not intersect with either of the lines, there is no intersection with C_1 and the step size is set to ∞.

explain what happens if lambda is zero or very small, or if lines are almost parallel
explain the geometric meaning of infinity case

If reaction i in $\tilde{x}$ has $v_i < 0$ and $\Delta\mu_i < \epsilon$, the intersection between the conic relaxation and C_2 is computed analogously. The equations to solve are $(\tilde{\boldsymbol{x}} + \lambda_1 \mathbf{r}^j)_{k_1} = \mathbf{0}$ and $(\tilde{\boldsymbol{x}} + \lambda_1 \mathbf{r}^j)_{k_2} = \boldsymbol{\epsilon}$.

write about possible usage of intersection cut

write about conic relaxation extraction in detail

example

4.7 Disjunctive Programming

Problem (40) written as disjunctive program is:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{50a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{50b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{50c}$$

$$\boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal \mathbf{S}_{\mathcal{I}} \tag{50d}$$

$$\begin{bmatrix} Y_i \\ v_i \geq 0 \\ \Delta\mu_i \leq -\epsilon \end{bmatrix} \vee \begin{bmatrix} Y_{i+|\mathcal{I}|} \\ v_i \leq 0 \\ \Delta\mu_i \geq \epsilon \end{bmatrix} \quad \forall i \in \mathcal{I} \tag{50e}$$

$$Y_i \vee Y_{i+|\mathcal{I}|} \quad \forall i \in \mathcal{I} \tag{50f}$$

where we have Boolean variables $Y_i \in \{true, false\}$ in addition to the continuous variables $\mathbf{v} \in \mathbb{R}^n, \boldsymbol{\Delta\mu} \in \mathbb{R}^{|(\mathcal{I})|}, \boldsymbol{\mu} \in \mathbb{R}^m$.

The DisjunctiveProgramming package extends JuMP and enables formulating disjunctive programs and provides several MIP reformulations [34] . The package is used to model Problem (50) and to acquire the big-M reformulation and the hull reformulation (see Section 2.4). For both reformulations the Boolean variable Y_i is modeled by a binary variable y_i. Constraint (50f) is then:

$$y_i + y_{i+|\mathcal{I}|} = 1 \quad \forall i \in \mathcal{I}$$

The big-M reformulation of Constraint (50e) is:

$$\begin{aligned} -v_i &\leq M(1 - y_i) \\ v_i &\leq M(1 - y_{i+|\mathcal{I}|}) \\ \Delta\mu_i &\leq -\epsilon + M(1 - y_i) \\ -\Delta\mu_i &\leq -\epsilon + M(1 - y_{i+|\mathcal{I}|}) \end{aligned}$$

which is identical to Problem (45) except that the big-M reformulation of Problem (50) uses two binary variables per disjunction instead of one. The tightness of the relaxed feasible solution of the big-M reformulation depends on the scalar M. The package finds the optimal value of M by using interval arithmetic [1].

For the hull reformulation, two continuous variables are added for each decision variable in a disjunction. The variables corresponding to v_i are denoted as v_{i_1} and v_{i_2} and the variables corresponding to $\Delta\mu_i$ are $\Delta\mu_{i_1}$ and $\Delta\mu_{i_2}$. The hull reformulation

of Constraint (50e) is:

$$
\begin{aligned}
& v_i = v_{i_1} + v_{i_2} \\
& \Delta\mu_i = \Delta\mu_{i_1} + \Delta\mu_{i_2} \\
& v_{i_1} \leq 0 \\
& v_{i_2} \leq 0 \\
& \Delta\mu_{i_1} \leq -y_i \\
& -\Delta\mu_{i_2} \leq -y_{i+|\mathcal{I}|} \\
& y_i + y_{i+|\mathcal{I}|} = 1 \\
& 0 \leq v_{i_j} \leq M v_{i_j} \quad \forall j \in \{1, 2\}
\end{aligned}
$$

We see that the hull reformulation needs more decision variables and constraints than the big-M reformulation. However, the feasible region of the hull reformulation is the convex hull of the disjunctions and therefore the tightest possible convex approximation.

4.8 Models

4.8.1 BiGG

We use a subset of metabolic networks of the *biochemical, genetic, and genomic* (BiGG) database [22]. The models selected cover the different model sizes the smallest being e_coli_core and the largest Recon3D. Table 1 lists the models including the number of metabolites and the number of reactions.

4.8.2 Yeast

4.8.3 GECKO

When adding enzyme constraints to the model, the flux rate depends on the enzymes and the flux bounds $\mathbf{l}, \mathbf{u}$ are only required to restrict the direction of a reaction. We use COBREXA to build the enzyme models, and we generate the enzyme data randomly. The turnover numbers differentiate for the forward and backward direction of a reaction and therefore we split each reversible reactions into one forward and one backward reaction. At least one enzyme is mapped to each reaction including the turnover numbers for the reaction in the forward and the turnover number for the backward direction are defined. The turnover numbers are taken from a normal distribution. The protein concentrations have to be in the interval $[0, 1000]$. An enzyme is made up of one or more proteins. Each protein is randomly associated

with either mass group A or B, and the product mass of each group is bounded by 0.5 mml/gDW. For each protein we assign a product molar mass randomly from a Uniform distribution. The stoichiometric matrix including enzyme data $\mathbf{S}^{GECKO}$ has a row for each metabolite and for each protein in the nework. The columns are the metabolic reactions in addition to an enzyme balance constraint linking each isozyme to its protein compound and the reaction it catalyzes.

enzymes are protein compounds

The size of $\mathbf{S}^{GECKO}$ is therefore much larger than $\mathbf{S}$. As an example, the stoichiometric matrix of e_coli_core including enzyme data is of size $\mathbb{R}^{209\times334}$. The number of rows results from 72 metabolites and 137 proteins, and we have 334 columns, instead of 95.

organisms in textsf?

organism	# metabolites	# reactions
e_coli_core	72	95
iAB_RBC_283	342	469
iIS312_Amastigote	606	519
iAF692	628	690
iSB619	655	743
iNF517	650	754
iHN637	698	785
iJB785	768	849
iNJ661	825	1025
iSynCJ816	928	1044
iJN746	907	1054
iJR904	761	1075
iEK1008	998	1226
iCN900	885	1229
iYO844	990	1250
iND750	1059	1266
iMM904	1226	1577
iRC1080	1706	2191
iAF1260	1668	2382
iSDY_1059	1888	2539
STM_v1_0	1802	2545
iJO1366	1805	2583
iSbBS512_1146	1910	2591
iS_1188	1914	2619
iSFV_1184	1917	2621
iSF_1195	1917	2630
iSFxv_1172	1918	2638
iML1515	1877	2712
iZ_1308	1923	2721
iAPECO1_1312	1942	2735
iECB_1328	1951	2748
iETEC_1333	1962	2756
iYS1720	2436	3357
iMM1415	2775	3726
RECON1	2766	3741
iLB1027_lipid	2172	4456
Recon3D	5835	10600

Table 1: Model size of used BiGG models.

5 Results

how to handle suboptimally solved instances

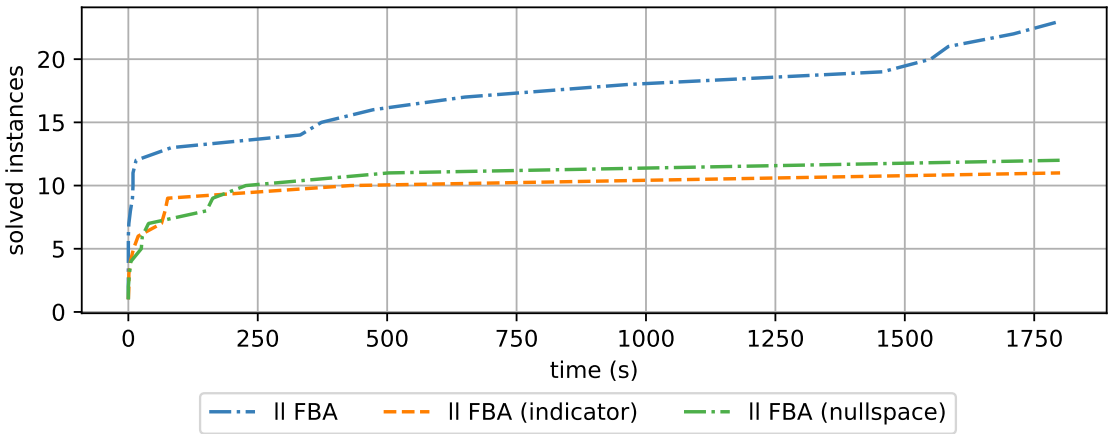
font for BiGG, SCIP, ll-FBA etc?

In this section we present the results of the computational experiments. The performance of the baseline model is shown in Section 5.1. We experiment with blocking cycles, no-good cuts, combinatorial Benders' cuts and disjunctive programming.

5.1 FBA and ll-FBA Variants

First, we look at the performance of the different ll-FBA variants. We compare the running time and number of solved BiGG instances of ll-FBA (indicator) (Problem (44)), of ll-FBA (Problem (45)) and of ll-FBA (nullspace) Problem (45) with the nullspace constraint (see Section 4.2).
The results are shown in Figure 12. ll-FBA solves around 60% of instances, whereas ll-FBA (nullspace) solves around 30%. With ll-FBA (indicator) we solve around 30% of the instances. Appendix Table 1 shows that for iMM1415 and iJN746, ll-FBA with indicator formulation solves much faster than with big-M formulation.

Figure 12: performance of ll-FBA variants



The results are shown in Appendix Table 1 and in Appendix Table 2

update label to "ll-FBA", change to latex font

For the other experiments, we only look at Problem (44) and Problem (45)

explain why

.

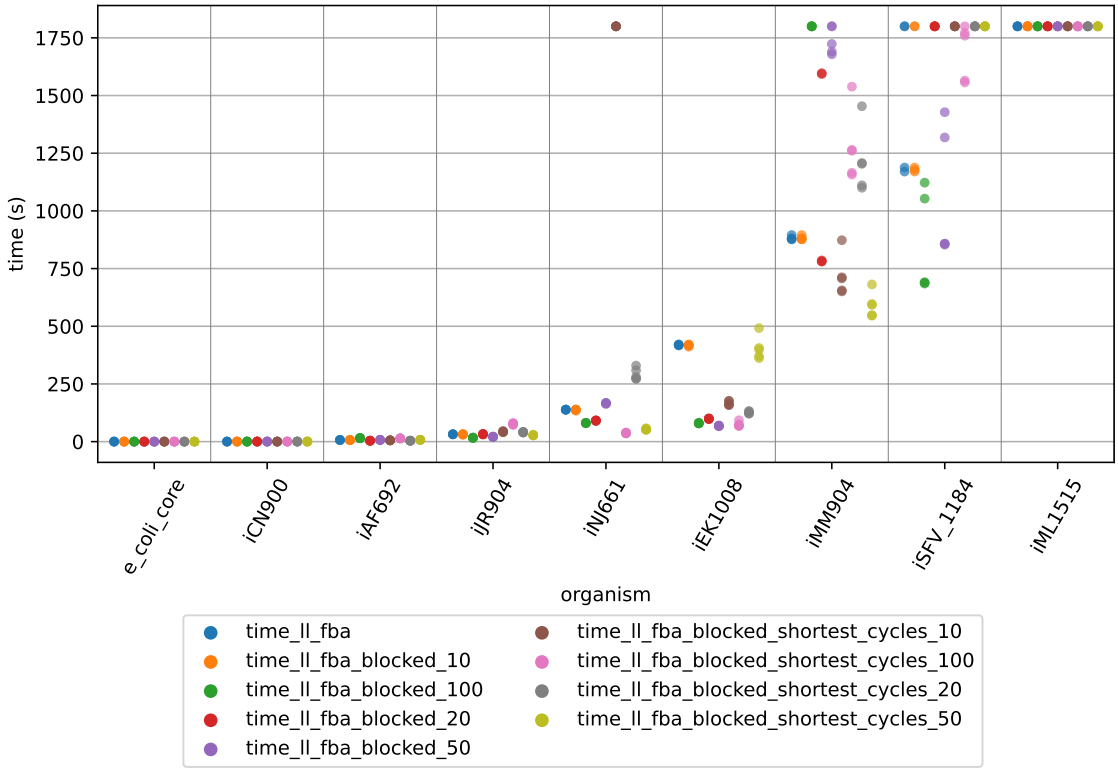
5.2 Blocking Cycles in FBA

We experiment with blocking cycles in ll-FBA (Problem (45)) as explained in Section 4.3 on a subset of BiGG instances. The cycle search is limited by 600 minutes.

check running time of instances

We then compare ll-FBA to 8 different setups. We block any 10, 20, 50 and 100 cycles prior to solving ll-FBA, and we block the same number of cycles by adding the shortest cycles detected. We solve each setup with five different SCIP seeds for each organism. The results are shown in Figure 13.

Figure 13: running time of ll-FBA and ll-FBA with blocked cycles



The results are shown in Appendix Table 3 and Appendix Table 4.

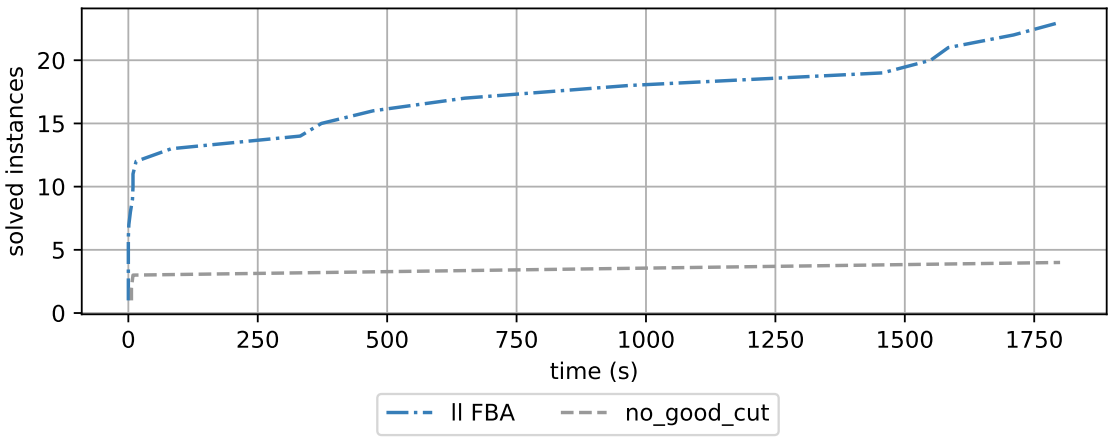
remove "time_" from legend, use latex font, change order of hues

mention small instance variability

5.3 Cutting Planes

We first present a comparison of Problem (45) with no-good cuts (see Section 4.4). While Problem (45) solves around 60%, we solve only 8% with no-good cuts. Figure 14 shows the running time and the number of solved instances.
As with no-good cuts, we only block one assignment of binary variables **a** per iteration, it requires many cuts to arrive at the integer hull.

Figure 14: runtime of ll-FBA and ll-FBA with no-good cuts



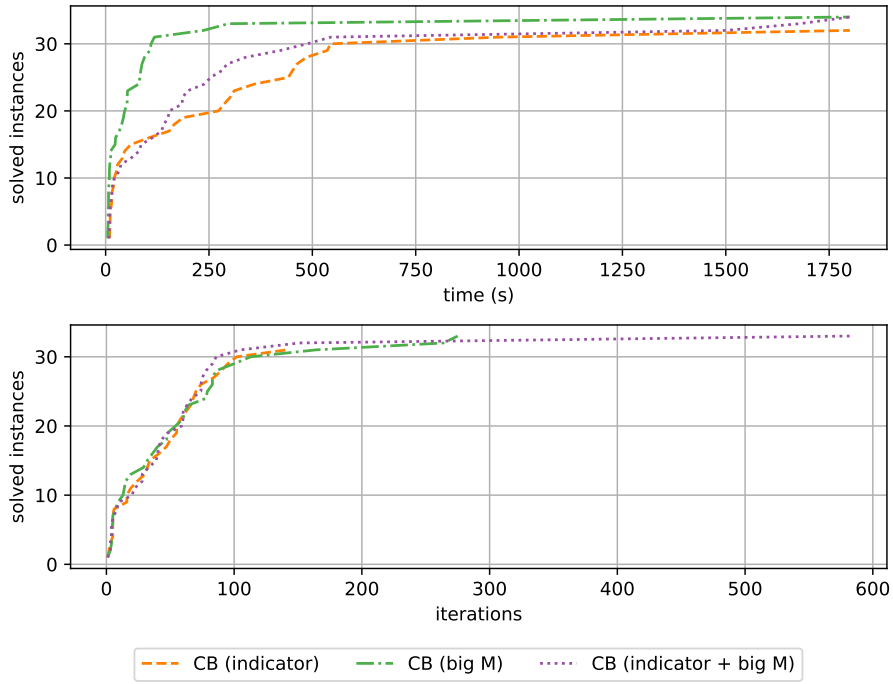
The results are shown in Appendix Table 5

Next, we look at the comparison of the ll-FBA variants with combinatorial Benders' cuts. We experiment with different formulations of the master problem and add one cut per iteration. Figure 15 shows ...

more detail on MP

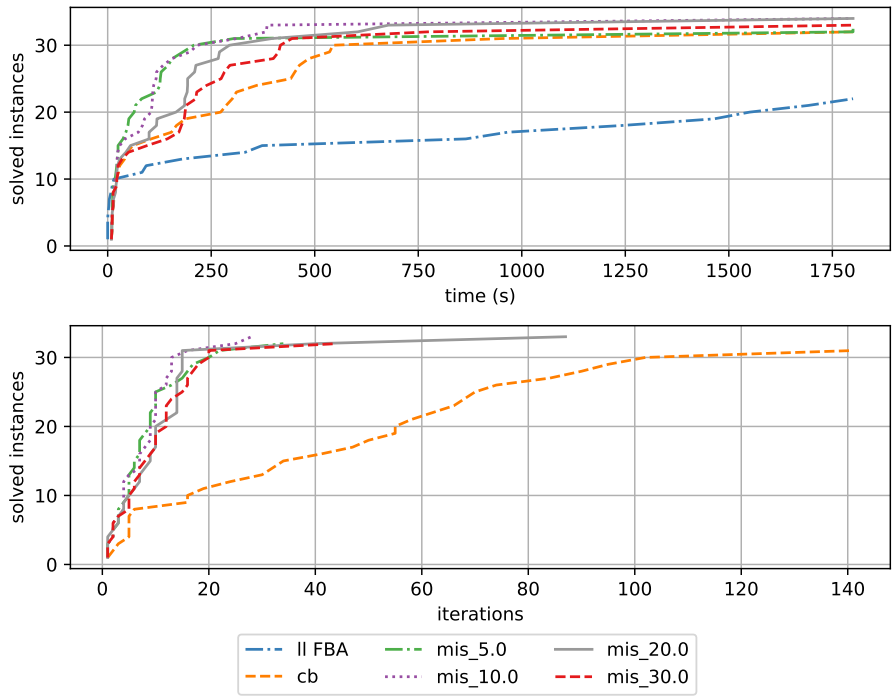
add ll FBA to plot

Figure 15: solved instances by CB (big-M), CB (indicator) and CB (indicator and big-M)



Next, we look at the comparison of Problem (44) with combinatorial Benders' cuts (see Section 4.5). In Figure 16...

Figure 16: solved instances by CB (indicator) and ll-FBA

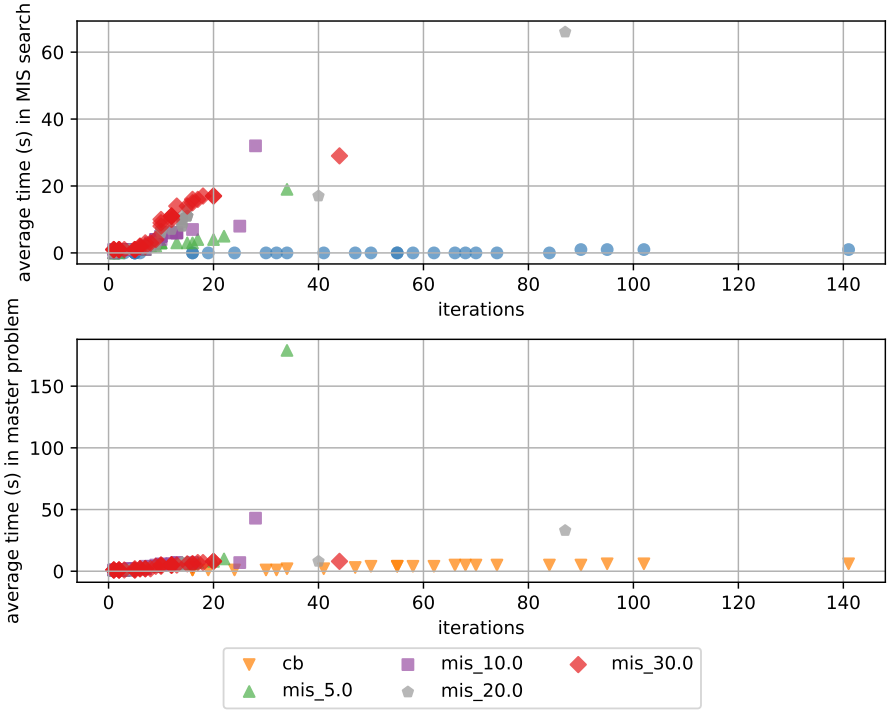


In Figure 17, we see that with several cuts per iteration, the number of iterations is

reduced while the time spent in the MIS search is larger.

check blue points, not in legend

Figure 17: solved instances by CB (indicator) and ll-FBA



Next, we compare Problem (45) with combinatorial Benders' cuts (see Section 4.5) by experimenting with the number of cuts added per iteration.

detail on MIS, meaning of label

add less cuts

note on yeast models

Figure 18: solved instances by CB (big-M) and ll-FBA

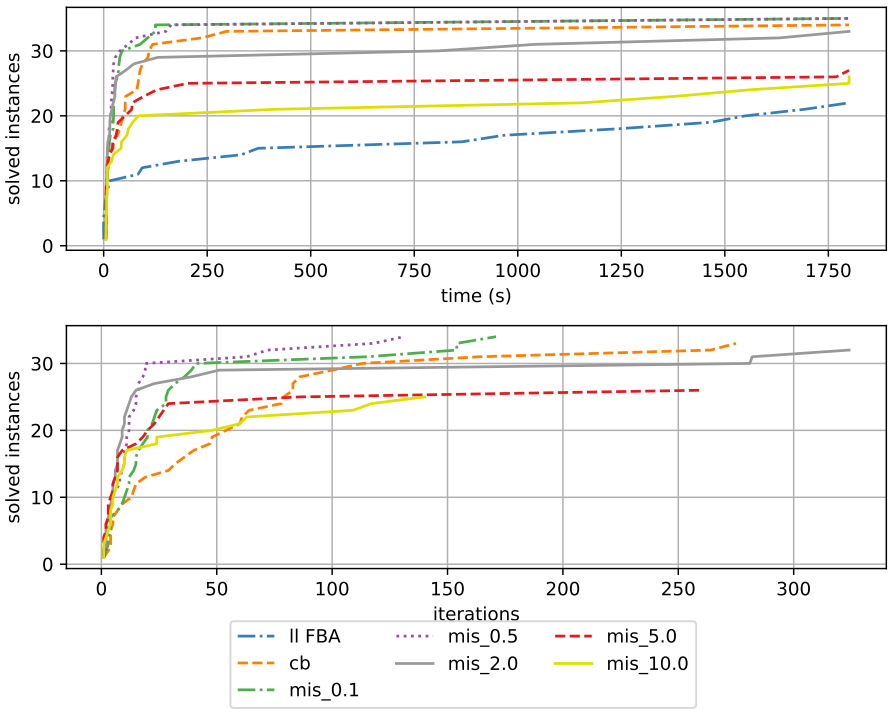


Figure 19: solved instances by CB (big-M) and ll-FBA

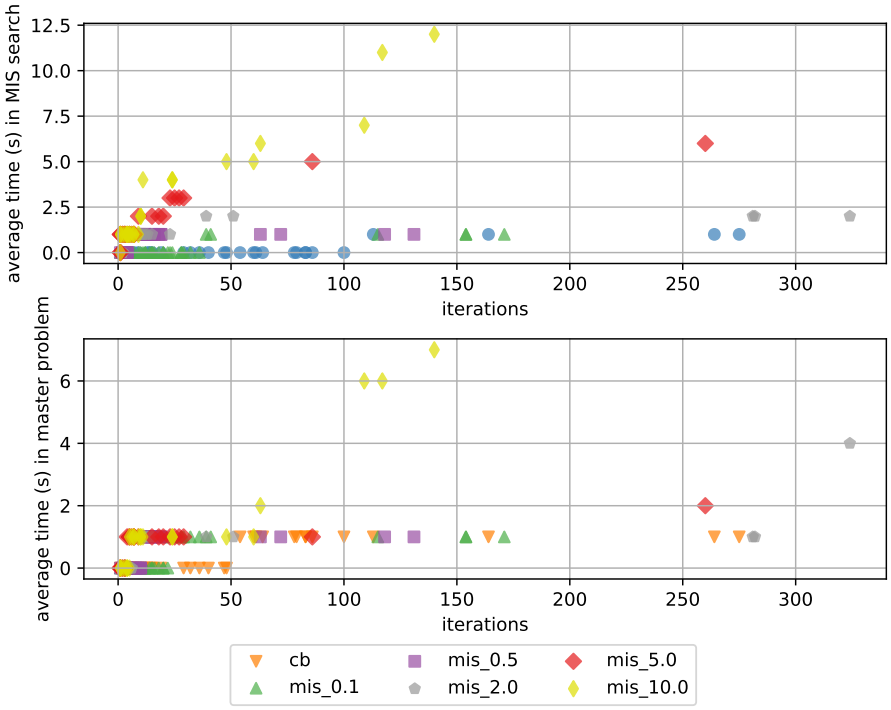
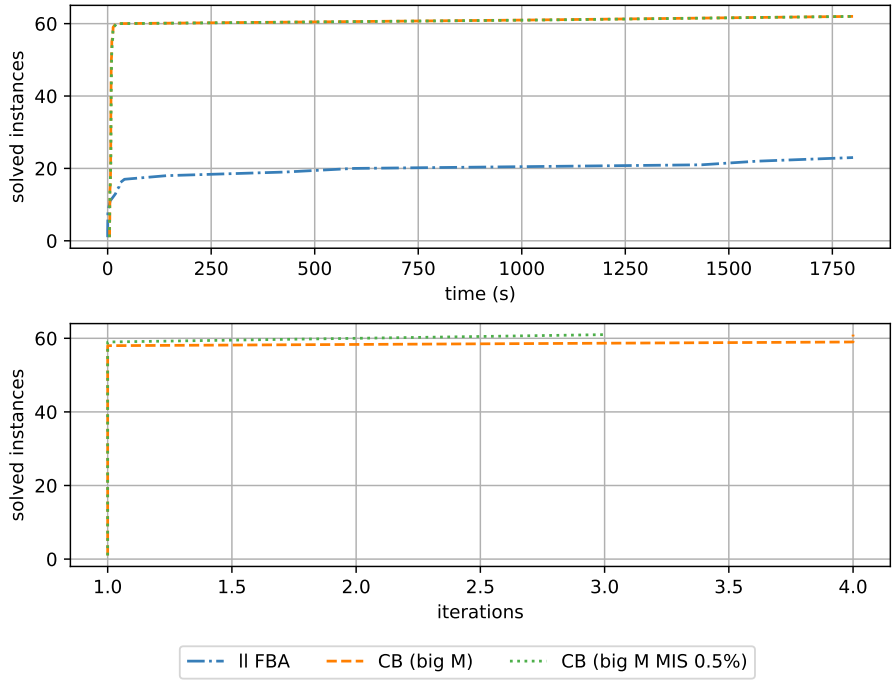
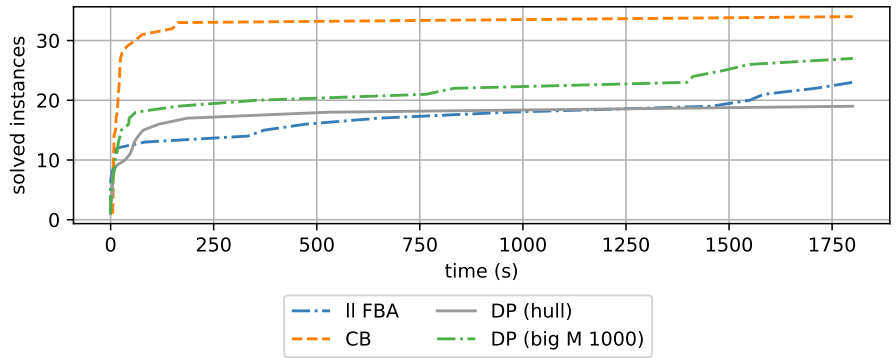


Figure 20: solved GECKO instances by CB (big-M), CB (big-M MIS 5%) and ll-FBA



5.4 Disjunctive Programming

Figure 21: solved instances by DP.jl, ll-FBA and CB



instances with smaller objective value are considered as NOT SOLVED, update plot

6 Conclusion

do not enumerate single subsections, only multiple

use AE or BE consistently

align all problems that are on the same page

bold a should look like normal a in bold

page with list of symbols and abbreviations

add var types to mathematical programs

use big-M with vector or scalar

use nonzero

polyhedral P, vs primal $\mathbb{P}$ vs optimization problem P, $\mathcal{P}$

program vs problem

fix cover page, header

fix page number of bibliography, fix bib entries

References

[1] A. Agarwal, S. Bhat, A. Gray, and I. E. Grossmann. Automating mathematical program transformations. In M. Carro and R. Peña, editors, *Practical Aspects of Declarative Languages*. Volume 5937, pages 134–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN: 978-3-642-11502-8 978-3-642-11503-5. DOI: 10.1007/978-3-642-11503-5_12. Series Title: Lecture Notes in Computer Science.

[2] M. ApS. MOSEK modeling cookbook.

[3] E. Balas. *Disjunctive Programming*. Springer International Publishing, Cham, 2018. DOI: 10.1007/978-3-030-00148-3. URL: http://link.springer.com/10.1007/978-3-030-00148-3 (visited on 05/19/2023).

[4] J. R. Banga. Optimization in computational systems biology. *BMC Systems Biology*, 2(1):47, May 28, 2008. ISSN: 1752-0509. DOI: 10.1186/1752-0509-2-47. URL: https://doi.org/10.1186/1752-0509-2-47 (visited on 03/04/2024).

[5] M. Basan, S. Hui, H. Okano, Z. Zhang, Y. Shen, J. R. Williamson, and T. Hwa. Overflow metabolism in escherichia coli results from efficient proteome allocation. *Nature*, 528(7580):99–104, Dec. 2015. DOI: 10.1038/nature15765.

[6] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. v. Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. v. d. Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, Dec. 2021. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

[7] D. Bienstock, C. Chen, and G. Muñoz. Outer-product-free sets for polynomial optimization and oracle-based cuts. *Mathematical Programming*, 183(1):105–148, Sept. 1, 2020. ISSN: 1436-4646. DOI: 10.1007/s10107-020-01484-3. URL: https://doi.org/10.1007/s10107-020-01484-3 (visited on 10/23/2023).

[8] S. Boyd and V. Lieven. *Convex Optimization*. Cambridge University Press, 2004. URL: https://web.stanford.edu/~boyd/cvxbook/ (visited on 06/22/2023).

[9] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006. ISSN: 0030-364X. URL: https://www.jstor.org/stable/25147009 (visited on 06/06/2023). Publisher: INFORMS.

[10] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer Programming*, volume 271 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham, 2014. DOI: 10.1007/978-3-319-11008-0. URL: https://link.springer.com/10.1007/978-3-319-11008-0 (visited on 11/13/2023).

[11] Constraint-based reconstruction and exascale analysis. URL: https://github.com/LCSB-BioCore/COBREXA.jl (visited on 02/15/2024).

[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN: 0262033844.

[13] A. Cornelis. Metabolic networks, thermodynamic constraints, and matroid theory.

[14] S. H. D. Dadush. Smoothed analysis of the simplex method. In *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021. DOI: 10.1017/9781108637435.

[15] A. A. Desouki, F. Jarre, G. Gelius-Dietrich, and M. J. Lercher. CycleFreeFlux: efficient removal of thermodynamically infeasible loops from flux distributions. *Bioinformatics*, 31(13):2159–2165, July 1, 2015. ISSN: 1367-4811, 1367-4803. DOI: 10.1093/bioinformatics/btv096. URL: https://academic.oup.com/bioinformatics/article/31/13/2159/195895 (visited on 08/02/2023).

[16] Dualization.jl. URL: https://github.com/jump-dev/Dualization.jl (visited on 02/15/2024).

[17] S. Even, A. Itai, and S. Shamir. On the complexity of time table and multi-commodity flow problems. Technical report 59, Technion, Department of Computer Science, 1975.

[18] J. Fairbanks, M. Besançon, S. Simon, J. Hoffiman, N. Eubank, and S. Karpinski. Juliagraphs/graphs.jl: an optimized graphs package for the julia programming language, 2021. URL: https://github.com/JuliaGraphs/Graphs.jl/.

[19] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177–201, Apr. 1993. ISSN: 0166218X. DOI: 10.1016/0166-218X(93)90045-P. URL: https://linkinghub.elsevier.com/retrieve/pii/0166218X9390045P (visited on 06/07/2023).

[20] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, Mar. 1, 2018. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5. URL: https://doi.org/10.1007/s12532-017-0130-5 (visited on 01/26/2024).

[21] S. Huiberts. *How Large is the Shadow?* English. PhD thesis, Mathematical Sciences at Utrecht University, 2017.

[22] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson, and N. E. Lewis. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44:D515–D522, D1, Jan. 4, 2016. ISSN: 0305-1048. DOI: 10.1093/nar/gkv1049. URL: https://doi.org/10.1093/nar/gkv1049 (visited on 01/18/2024).

[23] B. Legat, O. Dowson, J. Dias Garcia, and M. Lubin. MathOptInterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689, 2021. DOI: 10.1287/ijoc.2021.1067.

[24] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992. DOI: https://doi.org/10.1016/0020-0190(92)90208-D.

[25] M. Lubin, O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023. DOI: 10.1007/s12532-023-00239-3.

[26] A. C. Müller and A. Bockmayr. Fast thermodynamically constrained flux variability analysis. *Bioinformatics*, 29(7):903–909, Apr. 1, 2013. ISSN: 1367-4811, 1367-4803. DOI: 10.1093/bioinformatics/btt059. URL: https://academic.oup.com/bioinformatics/article/29/7/903/253327 (visited on 06/12/2023).

[27] F. S. Musalem. On cutting planes for mixed-integer nonlinear programming.

[28] E. Noor. Removing both internal and unrealistic energy-generating cycles in flux balance analysis, Mar. 13, 2018. arXiv: 1803.04999[q-bio]. URL: http://arxiv.org/abs/1803.04999 (visited on 08/08/2023).

[29] E. Noor, N. E. Lewis, and R. Milo. A proof for loop-law constraints in stoichiometric metabolic networks. *BMC Systems Biology*, 6(1):140, Nov. 12, 2012. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-140. URL: https://doi.org/10.1186/1752-0509-6-140 (visited on 05/12/2023).

[30] *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5. URL: http://link.springer.com/10.1007/978-0-387-40065-5 (visited on 10/12/2023).

[31] J. D. Orth, I. Thiele, and B. Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, Mar. 2010. ISSN: 1546-1696. DOI: 10.1038/nbt.1614. URL: https://www.nature.com/articles/nbt.1614 (visited on 04/25/2023). Number: 3 Publisher: Nature Publishing Group.

[32] B. Ø. Palsson. *Systems Biology: Constraint-based Reconstruction and Analysis*. Cambridge University Press, Cambridge, 2015. ISBN: 978-1-107-03885-1. DOI: 10.1017/CBO9781139854610. URL: https://www.cambridge.org/core/books/systems-biology/7F8445BC87019806B3625DFC4B5C27D4 (visited on 03/15/2024).

[33] A. Passi, J. D. Tibocha-Bonilla, M. Kumar, D. Tec-Campos, K. Zengler, and C. Zuniga. Genome-scale metabolic modeling enables in-depth understanding of big data. *Metabolites*, 12(1):14, Dec. 24, 2021. ISSN: 2218-1989. DOI: 10.3390/metabo12010014. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8778254/ (visited on 01/18/2024).

[34] H. D. Perez, S. Joshi, and I. E. Grossmann. DisjunctiveProgramming.jl: generalized disjunctive programming models and algorithms for JuMP, Mar. 8, 2023. arXiv: 2304.10492[cs]. URL: http://arxiv.org/abs/2304.10492 (visited on 01/10/2024).

[35] N. D. Price, I. Famili, D. A. Beard, and B. Ø. Palsson. Extreme pathways and kirchhoff's second law. *Biophysical Journal*, 83(5):2879–2882, Nov. 2002. ISSN: 00063495. DOI: 10.1016/S0006-3495(02)75297-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0006349502752971 (visited on 08/07/2023).

[36] K. Raman. *An Introduction to Computational Systems Biology: Systems-Level Modelling of Cellular Networks*. Chapman and Hall/CRC, New York, May 31, 2021. 358 pages. ISBN: 978-0-429-48695-1. DOI: 10.1201/9780429486951.

[37] K. Raman and N. Chandra. Flux balance analysis of biological systems: applications and challenges. *Briefings in Bioinformatics*, 10(4):435–449, July 1, 2009. ISSN: 1467-5463. DOI: 10.1093/bib/bbp011. URL: https://doi.org/10.1093/bib/bbp011 (visited on 01/17/2024).

[38] B. J. Sánchez, C. Zhang, A. Nilsson, P.-J. Lahtvee, E. J. Kerkhoven, and J. Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. *Molecular Systems Biology*, 13(8):935, Aug. 3, 2017. ISSN: 1744-4292. DOI: 10.15252/msb.20167411.

[39] J. Schellenberger, N. E. Lewis, and B. Ø. Palsson. Elimination of thermodynamically infeasible loops in steady-state metabolic models. *Biophysical Journal*, 100(3):544–553, Feb. 2, 2011. ISSN: 0006-3495. DOI: 10.1016/j.bpj.2010.12.3707. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3030201/ (visited on 04/25/2023).

[40] M. Terzer. *Large scale methods to enumerate extreme rays and elementary modes*. Doctoral Thesis, ETH Zurich, 2009. DOI: 10.3929/ethz-a-005945733. URL: https://www.research-collection.ethz.ch/handle/20.500.11850/151568 (visited on 01/26/2024). Accepted: 2017-06-13T09:43:51Z.

[41] M. Turner, T. Koch, F. Serrano, and M. Winkler. Adaptive cut selection in mixed-integer linear programming. *Open Journal of Mathematical Optimization*, 4:1–28, July 20, 2023. ISSN: 2777-5860. DOI: 10.5802/ojmo.25. URL: https://ojmo.centre-mersenne.org/articles/10.5802/ojmo.25/ (visited on 12/05/2023).

[42] *Understanding and Using Linear Programming*. Universitext. Springer, Berlin, Heidelberg, 2007. DOI: 10.1007/978-3-540-30717-4. URL: http://link.springer.com/10.1007/978-3-540-30717-4 (visited on 11/07/2023).

Appendix

	ll-FBA			ll-FBA (nullspace)		
organism	**termination**	**time**	**objective value**	**termination**	**time**	**objective value**
e_coli_core	OPTIMAL	0	0.874	OPTIMAL	0	0.874
iAB_RBC_283	OPTIMAL	0	2.936	OPTIMAL	2	2.936
iIS312_Amastigote	OPTIMAL	0	25.339	OPTIMAL	0	25.339
iAF692	OPTIMAL	0	0.0	TIME_LIMIT	1800	0.026
iSB619	OPTIMAL	0	0.0	TIME_LIMIT	1800	0.027
iNF517	OPTIMAL	9	0.043	OPTIMAL	39	0.043
iHN637	OPTIMAL	4	0.224	OPTIMAL	5	0.224
iJB785	OPTIMAL	15	0.054	OPTIMAL	25	0.0
iNJ661	TIME_LIMIT	1800	0.014	OPTIMAL	227	0.053
iSynCJ816	OPTIMAL	1	0.0	INFEASIBLE	34	-
iJN746	OPTIMAL	332	1.4	TIME_LIMIT	1800	-
iJR904	OPTIMAL	9	0.0	OPTIMAL	163	0.922
iEK1008	OPTIMAL	473	0.0	OPTIMAL	503	0.058
iCN900	OPTIMAL	0	0.0	OPTIMAL	27	0.0
iYO844	TIME_LIMIT	1800	0.115	TIME_LIMIT	1800	0.0
iND750	OPTIMAL	8	0.0	OPTIMAL	151	0.0
iMM904	OPTIMAL	650	0.277	TIME_LIMIT	1800	0.0
iRC1080	OPTIMAL	373	0.0	TIME_LIMIT	1800	-
iAF1260	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iSDY_1059	OPTIMAL	1457	0.938	TIME_LIMIT	1800	0.0
STM_v1_0	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iJO1366	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iSbBS512_1146	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iS_1188	TIME_LIMIT	1800	0.832	TIME_LIMIT	1800	0.0
iSFV_1184	OPTIMAL	1710	0.894	TIME_LIMIT	1800	0.0
iSF_1195	OPTIMAL	1584	0.915	TIME_LIMIT	1800	-0.0
iSFxv_1172	OPTIMAL	966	0.894	TIME_LIMIT	1800	0.0
iML1515	TIME_LIMIT	1800	0.861	TIME_LIMIT	1800	-
iZ_1308	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iAPECO1_1312	TIME_LIMIT	1800	0.934	TIME_LIMIT	1800	0.0
iECB_1328	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iETEC_1333	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iYS1720	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iMM1415	OPTIMAL	1550	0.0	TIME_LIMIT	1800	-
RECON1	OPTIMAL	83	0.0	TIME_LIMIT	1800	-
iLB1027_lipid	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
Recon3D	TIME_LIMIT	1800	-	TIME_LIMIT	1800	-

Appendix Table 1: Results of using the nullspace formulation vs the reformulation.

check iJB785

add number of blocked cycles, correspond to block limit except for e coli core

	ll-FBA (big-M)			ll-FBA (indicator)		
organism	**termination**	**time**	**objective value**	**termination**	**time**	**objective value**
e_coli_core	OPTIMAL	0	0.874	OPTIMAL	0	0.874
iAB_RBC_283	OPTIMAL	0	2.936	OPTIMAL	1	2.936
iIS312_Amastigote	OPTIMAL	0	25.339	OPTIMAL	1	25.339
iAF692	OPTIMAL	0	0.0	TIME_LIMIT	1800	-
iSB619	OPTIMAL	0	0.0	INFEASIBLE	2	-
iNF517	OPTIMAL	9	0.043	OPTIMAL	19	0.043
iHN637	OPTIMAL	4	0.224	OPTIMAL	64	0.224
iJB785	OPTIMAL	15	0.054	OPTIMAL	429	0.054
iNJ661	TIME_LIMIT	1800	0.014	TIME_LIMIT	1800	0.0
iSynCJ816	OPTIMAL	1	0.0	INFEASIBLE	0	-
iJN746	OPTIMAL	332	1.4	OPTIMAL	10	1.4
iJR904	OPTIMAL	9	0.0	TIME_LIMIT	1800	0.0
iEK1008	OPTIMAL	473	0.0	TIME_LIMIT	1800	0.0
iCN900	OPTIMAL	0	0.0	OPTIMAL	4	0.0
iYO844	TIME_LIMIT	1800	0.115	TIME_LIMIT	1800	0.118
iND750	OPTIMAL	8	0.0	TIME_LIMIT	1800	0.0
iMM904	OPTIMAL	650	0.277	TIME_LIMIT	1800	-
iRC1080	OPTIMAL	373	0.0	TIME_LIMIT	1800	-
iAF1260	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iSDY_1059	OPTIMAL	1457	0.938	TIME_LIMIT	1800	-
STM_v1_0	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iJO1366	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iSbBS512_1146	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iS_1188	TIME_LIMIT	1800	0.832	TIME_LIMIT	1800	0.0
iSFV_1184	OPTIMAL	1710	0.894	TIME_LIMIT	1800	-
iSF_1195	OPTIMAL	1584	0.915	TIME_LIMIT	1800	-
iSFxv_1172	OPTIMAL	966	0.894	TIME_LIMIT	1800	-
iML1515	TIME_LIMIT	1800	0.861	TIME_LIMIT	1800	-
iZ_1308	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iAPECO1_1312	TIME_LIMIT	1800	0.934	TIME_LIMIT	1800	0.0
iECB_1328	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iETEC_1333	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iYS1720	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iMM1415	OPTIMAL	1550	0.0	OPTIMAL	71	0.0
RECON1	OPTIMAL	83	0.0	OPTIMAL	76	0.0
iLB1027_lipid	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
Recon3D	TIME_LIMIT	1800	-	TIME_LIMIT	1800	-

Appendix Table 2: Results of using the big-M formulation vs the indicator formulation.

add results for different SCIP seeds

error in iAF692

o.v. does not match for iAF692

ll-FBA cycles blocked				
block limit	10	20	50	100
organism	time (s)	time (s)	time (s)	time (s)
e_coli_core	0	0	0	0
iAF692	6	4	7	14
iCN900	0	0	0	0
iJR904	29	30	20	16
iEK1008	411	101	70	82
iMM904	880	781	1689	1800
iSFV_1184	1178	1800	854	685
iML1515	1800	1800	1800	1800
iNJ661	134	89	163	79

Appendix Table 3: Runtime of ll-FBA with cycles blocked. Block limit denotes the maximal number of blocked cycles. The cycles are computed by the CycleFreeFlux algorithm.

ll-FBA shortest cycles blocked				
block limit	10	20	50	100
organism	time (s)	time (s)	time (s)	time (s)
e_coli_core	0	0	0	0
iAF692	5	4	7	14
iCN900	0	0	0	0
iJR904	40	39	27	72
iEK1008	163	125	370	69
iMM904	656	1111	548	1165
iSFV_1184	1800	1800	1800	1565
iML1515	1800	1800	1800	1800
iNJ661	1800	280	53	37

Appendix Table 4: Runtime of ll-FBA with shortest cycles blocked. Block limit denotes the maximal number of blocked cycles. The cycles are computed by the CycleFreeFlux algorithm.

fix rounding when NaN in column

reference equations and design tables

add table for gecko results

add table for yeast results

add DP table

	ll-FBA		no-good cuts	
organism	**termination**	**time**	**termination**	**time**
e_coli_core	OPTIMAL	0	OPTIMAL	6
iAB_RBC_283	OPTIMAL	0	TIME_LIMIT	1800
iIS312_Amastigote	OPTIMAL	0	TIME_LIMIT	1800
iAF692	OPTIMAL	0	ERROR	-
iSB619	OPTIMAL	0	TIME_LIMIT	1800
iNF517	OPTIMAL	9	TIME_LIMIT	1800
iHN637	OPTIMAL	4	TIME_LIMIT	1800
iJB785	OPTIMAL	15	OPTIMAL	9
iNJ661	TIME_LIMIT	1800	TIME_LIMIT	1800
iSynCJ816	OPTIMAL	1	TIME_LIMIT	1800
iJN746	OPTIMAL	332	TIME_LIMIT	1800
iJR904	OPTIMAL	9	TIME_LIMIT	1800
iEK1008	OPTIMAL	473	TIME_LIMIT	1800
iCN900	OPTIMAL	0	TIME_LIMIT	1800
iYO844	TIME_LIMIT	1800	TIME_LIMIT	1800
iND750	OPTIMAL	8	OPTIMAL	6
iMM904	OPTIMAL	650	TIME_LIMIT	1800
iRC1080	OPTIMAL	373	TIME_LIMIT	1800
iAF1260	TIME_LIMIT	1800	TIME_LIMIT	1800
iSDY_1059	OPTIMAL	1457	TIME_LIMIT	1800
STM_v1_0	TIME_LIMIT	1800	TIME_LIMIT	1800
iJO1366	TIME_LIMIT	1800	TIME_LIMIT	1800
iSbBS512_1146	TIME_LIMIT	1800	TIME_LIMIT	1800
iS_1188	TIME_LIMIT	1800	TIME_LIMIT	1800
iSFV_1184	OPTIMAL	1710	TIME_LIMIT	1800
iSF_1195	OPTIMAL	1584	TIME_LIMIT	1800
iSFxv_1172	OPTIMAL	966	TIME_LIMIT	1800
iML1515	TIME_LIMIT	1800	TIME_LIMIT	1800
iZ_1308	TIME_LIMIT	1800	TIME_LIMIT	1800
iAPECO1_1312	TIME_LIMIT	1800	TIME_LIMIT	1800
iECB_1328	TIME_LIMIT	1800	TIME_LIMIT	1800
iETEC_1333	TIME_LIMIT	1800	TIME_LIMIT	1800
iYS1720	TIME_LIMIT	1800	TIME_LIMIT	1800
iMM1415	OPTIMAL	1550	TIME_LIMIT	1800
RECON1	OPTIMAL	83	TIME_LIMIT	1800
iLB1027_lipid	TIME_LIMIT	1800	TIME_LIMIT	1800
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 5: Results of ll-FBA and of solving the ll-FBA problem with no-good cuts.

	ll-FBA		CB (indicator)		CB (big-M)	
organism	termination	time	termination	time	termination	time
e_coli_core	OPTIMAL	0	OPTIMAL	9	OPTIMAL	5
iAB_RBC_283	OPTIMAL	0	OPTIMAL	11	OPTIMAL	5
iIS312_Amastigote	OPTIMAL	0	OPTIMAL	10	OPTIMAL	6
iAF692	OPTIMAL	3	OPTIMAL	16	OPTIMAL	7
iSB619	INFEASIBLE	1	OPTIMAL	15	OPTIMAL	6
iNF517	OPTIMAL	11	OPTIMAL	29	OPTIMAL	12
iHN637	OPTIMAL	4	OPTIMAL	11	OPTIMAL	7
iJB785	OPTIMAL	15	OPTIMAL	14	OPTIMAL	7
iNJ661	OPTIMAL	181	OPTIMAL	61	OPTIMAL	9
iSynCJ816	OPTIMAL	1	ERROR	-	ERROR	-
iJN746	OPTIMAL	332	OPTIMAL	41	OPTIMAL	237
iJR904	OPTIMAL	93	OPTIMAL	21	OPTIMAL	12
iEK1008	OPTIMAL	865	OPTIMAL	20	OPTIMAL	9
iCN900	OPTIMAL	0	OPTIMAL	11	OPTIMAL	6
iYO844	TIME_LIMIT	1800	OPTIMAL	26	OPTIMAL	8
iND750	OPTIMAL	8	OPTIMAL	46	OPTIMAL	10
iMM904	OPTIMAL	1465	OPTIMAL	154	OPTIMAL	24
iRC1080	OPTIMAL	373	TIME_LIMIT	1800	ERROR	-
iAF1260	TIME_LIMIT	1800	OPTIMAL	166	OPTIMAL	39
iSDY_1059	TIME_LIMIT	1800	OPTIMAL	273	OPTIMAL	46
STM_v1_0	TIME_LIMIT	1800	OPTIMAL	105	OPTIMAL	23
iJO1366	TIME_LIMIT	1800	OPTIMAL	286	OPTIMAL	52
iSbBS512_1146	TIME_LIMIT	1800	OPTIMAL	360	OPTIMAL	118
iS_1188	TIME_LIMIT	1800	OPTIMAL	311	OPTIMAL	52
iSFV_1184	OPTIMAL	1241	OPTIMAL	442	OPTIMAL	85
iSF_1195	TIME_LIMIT	1800	OPTIMAL	452	OPTIMAL	43
iSFxv_1172	OPTIMAL	966	OPTIMAL	301	OPTIMAL	53
iML1515	OPTIMAL	1692	OPTIMAL	188	OPTIMAL	33
iZ_1308	TIME_LIMIT	1800	OPTIMAL	536	OPTIMAL	83
iAPECO1_1312	TIME_LIMIT	1800	OPTIMAL	954	OPTIMAL	107
iECB_1328	TIME_LIMIT	1800	OPTIMAL	486	OPTIMAL	80
iETEC_1333	TIME_LIMIT	1800	OPTIMAL	544	OPTIMAL	88
iYS1720	TIME_LIMIT	1800	OPTIMAL	463	OPTIMAL	94
iMM1415	OPTIMAL	1550	TIME_LIMIT	1800	OPTIMAL	110
RECON1	OPTIMAL	83	TIME_LIMIT	1800	OPTIMAL	297
iLB1027_lipid	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 6: Results of using ll-FBA vs CB with indicator and CB with big-M formulation.

	CB (big-M) MIS 5%		CB (big-M) MIS 10%		CB (big-M) MIS 20%		CB (big-M
organism	termination	time	termination	time	termination	time	terminatio
e_coli_core	OPTIMAL	5	OPTIMAL	6	OPTIMAL	5	OPTIMAL
iAB_RBC_283	OPTIMAL	6	OPTIMAL	5	OPTIMAL	6	OPTIMAL
iIS312_Amastigote	OPTIMAL	6	OPTIMAL	6	OPTIMAL	6	OPTIMAL
iAF692	OPTIMAL	6	OPTIMAL	7	OPTIMAL	12	OPTIMAL
iSB619	OPTIMAL	6	OPTIMAL	9	OPTIMAL	8	OPTIMAL
iNF517	OPTIMAL	6	OPTIMAL	6	OPTIMAL	6	OPTIMAL
iHN637	OPTIMAL	6	OPTIMAL	7	OPTIMAL	9	OPTIMAL
iJB785	OPTIMAL	6	OPTIMAL	7	OPTIMAL	8	OPTIMAL
iNJ661	OPTIMAL	22	OPTIMAL	23	OPTIMAL	41	OPTIMAL
iSynCJ816	ERROR	-	ERROR	-	ERROR	-	ERROR
iJN746	OPTIMAL	128	OPTIMAL	84	OPTIMAL	192	OPTIMAL
iJR904	OPTIMAL	9	OPTIMAL	9	OPTIMAL	78	OPTIMAL
iEK1008	OPTIMAL	10	OPTIMAL	9	OPTIMAL	21	OPTIMAL
iCN900	OPTIMAL	6	OPTIMAL	9	OPTIMAL	10	OPTIMAL
iYO844	OPTIMAL	9	OPTIMAL	8	OPTIMAL	10	OPTIMAL
iND750	OPTIMAL	10	OPTIMAL	19	OPTIMAL	15	OPTIMAL
iMM904	OPTIMAL	33	OPTIMAL	70	OPTIMAL	146	OPTIMAL
iRC1080	ERROR	-	OPTIMAL	412	OPTIMAL	290	ERROR
iAF1260	OPTIMAL	15	OPTIMAL	42	OPTIMAL	89	OPTIMAL
iSDY_1059	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	OPTIMAL
STM_v1_0	OPTIMAL	52	TIME_LIMIT	1800	OPTIMAL	934	OPTIMAL
iJO1366	OPTIMAL	30	OPTIMAL	56	TIME_LIMIT	1800	TIME_LI
iSbBS512_1146	ERROR	-	ERROR	-	ERROR	-	ERROR
iS_1188	OPTIMAL	21	OPTIMAL	43	TIME_LIMIT	1800	TIME_LI
iSFV_1184	TIME_LIMIT	1800	OPTIMAL	60	OPTIMAL	42	TIME_LI
iSF_1195	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	OPTIMAL
iSFxv_1172	TIME_LIMIT	1800	OPTIMAL	1800	TIME_LIMIT	1800	TIME_LI
iML1515	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
iZ_1308	OPTIMAL	69	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
iAPECO1_1312	OPTIMAL	97	OPTIMAL	1559	TIME_LIMIT	1800	TIME_LI
iECB_1328	OPTIMAL	68	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
iETEC_1333	OPTIMAL	35	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
iYS1720	OPTIMAL	203	OPTIMAL	1155	TIME_LIMIT	1800	TIME_LI
iMM1415	OPTIMAL	1769	OPTIMAL	1380	TIME_LIMIT	1800	TIME_LI
RECON1	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
iLB1027_lipid	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LI

Appendix Table 7: Results of searching multiple MIS in CB with big-M formulation.

	CB (indicator) MIS 5%		CB (indicator) MIS 10%	
organism	termination_cb_mis_5	time_cb_mis_5	termination_cb_mis_10	time_cb_m
e_coli_core	OPTIMAL	9	OPTIMAL	9
iAB_RBC_283	OPTIMAL	10	OPTIMAL	10
iIS312_Amastigote	OPTIMAL	10	OPTIMAL	10
iAF692	OPTIMAL	14	OPTIMAL	14
iSB619	OPTIMAL	11	OPTIMAL	12
iNF517	OPTIMAL	12	OPTIMAL	16
iHN637	OPTIMAL	11	OPTIMAL	10
iJB785	OPTIMAL	12	OPTIMAL	11
iNJ661	OPTIMAL	23	OPTIMAL	29
iSynCJ816	ERROR	-	ERROR	-
iJN746	OPTIMAL	25	OPTIMAL	23
iJR904	OPTIMAL	18	OPTIMAL	22
iEK1008	OPTIMAL	20	OPTIMAL	23
iCN900	OPTIMAL	11	OPTIMAL	16
iYO844	OPTIMAL	22	OPTIMAL	18
iND750	OPTIMAL	37	OPTIMAL	24
iMM904	OPTIMAL	25	OPTIMAL	84
iRC1080	OPTIMAL	305	OPTIMAL	312
iAF1260	OPTIMAL	44	OPTIMAL	89
iSDY_1059	OPTIMAL	156	OPTIMAL	117
STM_v1_0	OPTIMAL	49	OPTIMAL	42
iJO1366	OPTIMAL	82	OPTIMAL	134
iSbBS512_1146	OPTIMAL	129	OPTIMAL	107
iS_1188	OPTIMAL	67	OPTIMAL	74
iSFV_1184	OPTIMAL	146	OPTIMAL	111
iSF_1195	OPTIMAL	64	OPTIMAL	108
iSFxv_1172	OPTIMAL	51	OPTIMAL	109
iML1515	OPTIMAL	126	OPTIMAL	385
iZ_1308	OPTIMAL	208	OPTIMAL	105
iAPECO1_1312	OPTIMAL	182	OPTIMAL	155
iECB_1328	OPTIMAL	127	OPTIMAL	199
iETEC_1333	OPTIMAL	118	OPTIMAL	118
iYS1720	ERROR	-	OPTIMAL	216
iMM1415	TIME_LIMIT	1800	TIME_LIMIT	1800
RECON1	TIME_LIMIT	1800	TIME_LIMIT	1800
iLB1027_lipid	OPTIMAL	1800	OPTIMAL	383
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 8: Results of searching multiple MIS in CB with indicator constraints.

	CB (big-M)			CH			CH MIS 5%		
organism	termination	time	o. v.	termination	time	o. v.	termination	time	o.
e_coli_core	OPTIMAL	5	0.874	OPTIMAL	6	0.874	OPTIMAL	5	0.
iAF692	OPTIMAL	7	0.013	OPTIMAL	10	0.027	OPTIMAL	10	0.
iNF517	OPTIMAL	12	0.043	OPTIMAL	28	0.043	OPTIMAL	19	0.
iNJ661	OPTIMAL	9	0.052	OPTIMAL	24	0.053	OPTIMAL	24	0.
iJR904	OPTIMAL	12	0.922	OPTIMAL	11	0.922	OPTIMAL	11	0.
iEK1008	OPTIMAL	9	0.058	OPTIMAL	36	0.058	OPTIMAL	36	0.
iCN900	OPTIMAL	6	0.0	OPTIMAL	5	0.0	OPTIMAL	6	0.
iMM904	OPTIMAL	24	0.287	OPTIMAL	75	0.288	OPTIMAL	76	0.
iAF1260	OPTIMAL	39	0.737	OPTIMAL	207	0.737	OPTIMAL	207	0.
iSDY_1059	OPTIMAL	46	0.938	OPTIMAL	201	0.938	OPTIMAL	204	0.
iJO1366	OPTIMAL	52	0.982	OPTIMAL	993	0.982	OPTIMAL	1039	0.
iSbBS512_1146	OPTIMAL	118	0.983	TIME_LIMIT	1800	0.978	TIME_LIMIT	1800	0.
iS_1188	OPTIMAL	52	0.857	OPTIMAL	225	0.857	OPTIMAL	226	0.
iSFV_1184	OPTIMAL	85	0.894	OPTIMAL	1156	0.894	OPTIMAL	1155	0.
iSF_1195	OPTIMAL	43	0.915	OPTIMAL	313	0.915	OPTIMAL	320	0.
iML1515	OPTIMAL	33	0.877	TIME_LIMIT	1800	0.877	TIME_LIMIT	1800	0.

Appendix Table 9: Results of using CB vs CH.